82SM-000011-1

パーソナルコンピュータ

FM-7

F-BASIC 文法書

パーソナルコンピュータ

F-BASIC文法書

富士通株式会社

お　願　い

1．本書を始めとする各種マニュアルについてのお問合わせは，お買上げの販売店，および「富士通マイコンスカイラブ」へお願いいたします．
2．本体および各種オプション品，周辺装置の取扱いについては，各種取扱説明書を充分にお読みのうえ，使用して下さい．
3．新製品ニュース，ソフトウェアについてのお知らせは，各種マイコン雑誌への広告および，保証書，アンケート用紙をご送付くださった方へのダイレクトメール等により行います．

富士通マイコンスカイラブ

虎ノ門：〒106　東京都港区虎ノ門２－３－13　第18森ビル内
TEL（03）591－1091，2561
月～金（祝日を除く）　9時30分～17時

秋葉原：〒101　東京都千代田区外神田１－15－16　秋葉原ラジオ会館６Ｆ
TEL（03）251－1448
年中無休　10時～19時

札　幌：〒060　札幌市中央区南一条西３丁目　丸井今井一条本館４Ｆ
TEL（011）241－4185
月～金（水曜定休日）　10時～18時
土・日　10時～18時30分

仙　台：〒980　仙台市国分町１－７－18　明治生命仙台国分町ビル１Ｆ
TEL（0222）66－8711

名古屋：〒460　名古屋市中区栄１－５－22　富士通ＯＡショールーム内
TEL（052）221－6016
月～土（祝日を除く）　10時～18時

大　阪：〒530　大阪市北区梅田１－２－２　大阪駅前第２ビル１Ｆ
TEL（06）344－7628
年中無休　10時～19時

広　島：〒733　広島市中区立町４－２　大橋ビル２Ｆ，３Ｆ
TEL（082）247－3949
年中無休　10時～19時

はじめに

本書は，パーソナルコンピュータ FUJITSU MICRO 7（略称 FM-7：エフ・エム・セブン），FUJITSU MICRO 8（略称 FM-8：エフ・エム・エイト）の標準プログラム言語である，F-BASIC（エフ・ベーシック）について解説したものです．

F-BASIC は，米国のマイクロソフト社の BASIC を大幅に拡張したものであり，FM-7，FM-8 のそれぞれの特長がいかされております．

本書が，BASIC のプログラミングにおいて，ユーザの皆様にお役に立つことを期待しております．

なお，本書は個々の命令についての解説が主体となっておりますので，操作法等につきましては，別冊の解説書をお読み下さい．

昭和 57 年 10 月

目　　次

第1章　F-BASIC

1.1 概　　要

「パーソナルコンピュータの標準的言語はBASICである.」と今では誰もがそう答えを出すほど, BASIC言語はポピュラーなものになってきました. BASIC (Beginner's All purpose Symbolic Instruction Code) 言語はもともとTSS (タイムシェアリング：Time Sharing System) から開発された言語であり, アメリカのマイクロソフト社がマイクロコンピュータ, パーソナルコンピュータの性能を生かすように作り直しました. FUJITSU MICRO 7 (略称FM-7：エフ・エム・セブン) とFUJITSU MICRO 8 (略称FM-8：エフ・エム・エイト) は, 標準言語としてBASICを採用し, さらにハードの特長を生かすために, いくつかの拡張機能を追加してF-BASIC (エフ・ベーシック) と命名し, 本体内に標準実装しました.

F-BASICは, 3つのバージョンがあり, V1.0, V2.0, V3.0と表記して区別します.

FM-8はV1.0, V2.0を, FM-7はV3.0をサポートしておりますが, この文法書は, バージョンによらず, すべての命令について解説しております.

1.2 F-BASICのバージョン

F-BASICはバージョンにより区別され次の表になります.

F-BASICのバージョン

バージョン	媒体	名称	略称	適用機種
V1.0	FM-8本体内のROM	F-BASIC V1.0 ROMモード	V1.0／ROM	FM-8
	FM-8本体内のROMおよびミニフロッピィディスクに添付されているシステムディスク	F-BASIC V1.0 DISKモード	V1.0／DISK	
V2.0	ミニフロッピィディスク	F-BASIC V2.0 5インチ版	V2.0／5	FM-8
	標準フロッピィディスク	F-BASIC V2.0 8インチ版	V2.0／8	
V3.0	FM-7本体内のROM	F-BASIC V3.0 ROMモード	V3.0／ROM	FM-7
	FM-7本体内のROMおよびV3.0用のシステムディスク	F-BASIC V3.0 DISKモード	V3.0／DISK	

(1) F-BASIC V1.0

F-BASIC V1.0はROMモードとDISKモードで構成されています.

(2) F-BASIC V2.0

F-BASIC V2.0はF-BASIC V1.0に以下に述べるコマンドや機能が追加されています.

〈追加機能〉

- CHAIN, COMMON, ERASE, LLIST, LPRINT, LPRINT USING文および LPOS関数の追加
- TERM文コマンドのエラー発生時の処理方法の変更
- PRINT USING文の書式制御文字の追加
- OPEN文でプリンタに対するオプション指定の追加
- ユーザプログラムの自動スタート機能

(3) F-BASIC V3.0

F-BASIC V3.0はF-BASIC V2.0に追加された機能と削除した機能があります.

〈追加機能〉

- アクティブ画面とディスプレイ画面を設定するSCREEN文の追加
- パレットコードを指定するCOLOR文の追加
- 音楽演奏機能を行うPLAY, SOUND文の追加

〈削除機能〉

- バブルカセットに対するBUBINI, BUBR, BUBW文の削除
- アナログ入力インタフェースに対するANPORT文の削除

第2章 F-BASICプログラミング上の規約

2.1 行 の 形 式

BASIC のプログラムは，行の集まりによって構成されます．行の形式は次のとおりです．

nnnnn BASIC のステートメント [：BASIC のステートメント…] [' コメント]

nnnnn は行番号を示し，5桁以内の10進数でなければなりません．

BASIC のステートメントには，実行文または非実行文を書くことができます．実行文は．BASIC がプログラムを実行するときに次に実行すべきものを通知するための命令です．これに対して非実行文は，BASIC の実行の制御がその文に渡ったときには，何ら動作をせず次の文にそのまま制御を渡します．例えばPRINT 文などは実行文であり，DATA，REM 文は非実行文になります．

1つの行には，複数個の BASIC のステートメントを書くことができます．このとき，各ステートメントはコロン（：）により，直前のステートメントと区切らなければなりません．

1つの行の終りには，コメントを付加することができますが，直前の BASIC のステートメントとは，シングルクォート（'）により分離しなければなりません．

1つの行はリターンキーの押下により終了しますが，一つの行のトータル文字数は255以内でなければなりません．

〔例〕
```
10 REM *** sample program ***
20 FOR I=1 TO 100:J=I*I:PRINT J:NEXT
30 END 'sample program END
```

2.2 行番号の参照

BASIC プログラムの全ての行は，行番号で始まります．行番号は，プログラムがメモリに格納されるときの順序及びプログラムの実行の順序を示します．プログラムの実行は，行番号の小さいものから順に行われます．

行番号は，また GOTO，GOSUB などのステートメント，あるいは LIST，DELETE，EDIT などのコマンドで参照することができます．行番号は0から63999の範囲の整数でなければなりません．

行番号の代りにピリオド（・）が使える場合があります．ピリオドは，LIST，AUTO，DELETE，

EDITなどのコマンドで，エラー発生，編集時における現在の行を表す行番号として使用できます．

〔例〕　LIST.
　　　　AUTO.
　　　　DELETE. －100
　　　　EDIT.

2.3 文字セット

F-BASIC言語で使用できる文字セットは，英字，数字，カナ文字及び特殊文字により構成されます．

英字は，AからZまでの26文字であり，それぞれ大文字と小文字があります．

数字は，0から9までの10文字です．

特殊文字には，特殊記号及びグラフィック文字があります．

以下の特殊記号は，BASICでは特別な意味を持つ文字として使われます．

文字	名称
	ブランク
=	等価記号または代入記号
+	正符号または加算記号
－	負符号またはハイフン
*	アスタリスクまたは乗算記号
/	スラッシュまたは除算記号
¥	円記号または整数除算記号
^	矢印またはベキ乗記号
(	左カッコ
)	右カッコ
%	パーセント
#	ナンバ記号
$	ドル記号
!	エクスクラメーションまたは感嘆符
&	アンパサント
@	アットマーク
,	コンマ
.	ピリオドまたは小数点
'	シングルクォート
;	セミコロン
:	コロン
?	疑問符またはクエスションマーク
<	より小さい
>	より大きい
"	ダブルクォーテーションまたは引用符
_	アンダスコア

2.4 定　　数

定数はそれ自身が値を表し，F-BASICで扱える定数には，文字定数と数値定数があります．

2.4.1 文 字 定 数

文字定数は，F-BASICで扱える文字セットを並べ，全体をダブルクォーテーション（″）で囲って指定します．この文字数の長さは，255以下でなければなりません．特に文字の長さが0の文字列を空文字列と言います．

〔例〕　"THANK YOU"
　　　　"ヒンメイ　タンカ　ウリアゲ"
　　　　""　　　　空文字列

2.4.2 数 値 定 数

数値定数は正または負の数です．BASICでは数値定数の間にコンマを含めることはできません．数値定数には，次の5つの形式があります．

（1） 整数形式

プラス（+）またはマイナス（−）の符号に続いて1つ以上の数字を並べたもので，プラス符号は省略することができます．

プラス符号の付いたものを正の整数，マイナス符号の付いたものを負の整数と呼び，整数の値の範囲は，−32768から+32767までです．

〔例〕　1
　　　　+123
　　　　−32767

（2） 固定小数点形式

符号に続いて，整数部，小数点，小数部の順に書いたもので，プラス符号は省略することができます．整数部と小数部のうち，一方は省略できますが，両方とも省略することはできません．小数点はピリオド（.）で表します．

〔例〕　1.0
　　　　−123.11
　　　　.999

（3） 浮動小数点形式

指数形式で表現された正または負の数値で，符号に続いて，整数部，小数点，小数部，指数部の順に書いたものです．プラス符号は省略することができます．また，整数部，小数部のうち，一方の省略あるいは，小数点と小数部の省略ができます．

指数部は，単精度では E [{±}] nn で表し，倍精度では D [{±}] nn で表します．なお，nn は符号なしの整数です．

〔例〕　５２０．１８Ｅ＋７
　　　　１０８Ｅ－１８
　　　　０．１２５６Ｅ＋１２
　　　　０．９９９９９９９９Ｄ－８

（４） 16進形式

プレフィックス &H に続いて，16進数（0～9，A～F）を並べて書いたものです．先行する 0 を除いて 4 桁まで書くことができ，&H0 から &HFFFF までの範囲を表すことができます．この形式で入力された16進数は，符号なし10進数に変換されて出力されます．

〔例〕　＆Ｈ６４
　　　　＆Ｈ００３Ｆ

（５） 8進形式

プレフィックス &O または，& に続いて 8 進数（0～7）を並べて書いたものです．先行する 0 を除いて 6 桁まで書くことができ，&O0～&O177777 までの範囲を表すことができます．この形式で入力された 8 進数は，符号なし10進数に変換されて出力されます．

〔例〕　＆Ｏ０１２３
　　　　＆Ｏ１２３４５６

単精度と倍精度の数値定数　数値定数には，単精度定数と倍精度定数があります．単精度定数は，7 桁までの精度で格納され，6 桁までの桁数で印刷されます．倍精度定数は，17桁の精度で格納され，16桁までの桁数で印刷されます．単精度定数および倍精度定数で表現できる数値の絶対値は，約 3.0×10^{-39} から $1.7\times10^{+38}$ までであり，3.0×10^{-39} より小さい数は 0 になります．

単精度数値定数は次のいずれかに当てはまるものです．

（１） 有効桁数が 7 桁以下の定数．
（２） E を用いた指数形式で表現されたもの．
（３） 最後に（！）が書かれた定数．

〔例〕　１３．６
　　　－１．２３Ｅ＋６
　　　　８６２．１！
　　　　５２０

倍精度数値定数は，次のいずれかに当てはまるものです．

（１） 有効桁数が 8 桁以上の定数．
（２） D を用いた指数形式で表現されたもの．
（３） 最後にナンバ記号（＃）が書かれた定数．

〔例〕　1368792702
　　　−1.5670D−12
　　　562.983#
　　　852.138269012

2.5 変　　数

変数は，BASICのプログラムの中で使用される値を表すために使われる名前です．定数と同じ様に，変数にも数値型（数値変数）と文字型（文字変数）の2つがあります．数値変数は常に数値を持っています．文字変数は文字ストリングだけを持っています．文字変数の値の長さは，一定ではありません．文字変数に値が代入されたとき，文字変数の長さが決まりますが，その長さは0から255の範囲になければなりません．

変数に値を与えるためには，代入文，INPUT文，READ文等を用いますが，いずれの場合にも変数の型（文字又は数値）は，それに割り当てられる定数の型と一致してなければなりません．

変数に値を与える前に数値変数を用いると，その値は0になっています．文字変数の場合はヌルストリングになっており，何の文字も持たない状態になっています．

2.5.1 変数名と型宣言文字

BASICの変数名は，それを用いたステートメントが1行以内に納まるならば，何文字であってもかまいません．もし変数名が16文字をこえているならば，BASICは最初の16文字と型宣言文字，（もし，付けられているならば）により変数名を区別します．変数名の中で許される文字は，英字及び数字ですが，最初の文字は英字でなければなりません．英字においては，大文字と小文字の区別はありません．英小文字で変数名を入力しても，英大文字に変換してメモリに格納されます．

変数名は予約語であったり，予約語で始めることはできません．

〔例〕　LIST　←　予約語であり変数としてなりません．
　　　LISTA←　LISTという予約語で始まっていますので変数としてなりません．
　　　ALIST←　変数として許されます．

予約語はBASICで使われるコマンド名，ステートメント名，関数名，演算子名などすべて予約語になります．詳細は「2.5.4 予約語」を参照して下さい．

変数は数値，文字列のいずれをも表すことができます．変数名の直後に型宣言文字を付けて，その変数が表す値の型を宣言することができます．この型宣言文字を用いることにより，変数名が同じであっても型が違うことにより別の変数として扱うことができます．型宣言文字には，次のものがあります．

%　整数変数を示し，1つの変数につき，2バイトのデータ格納域を必要とします．
!　単精度変数を示し，1つの変数につき，4バイトのデータ格納域を必要とします．
#　倍精度変数を示し，1つの変数につき，8バイトのデータ格納域を必要とします．
$　文字変数を示し，最大255文字のデータを格納することができます．

変数の型を宣言する方法として他に DEFINT, DEFSNG, DEFDBL, DEFSTR の型宣言文がありますが、これらについては「3.2.3」項を参照して下さい.

型宣言文もなく型宣言文字も伴わない変数名は，単精度形式の数値を表すものとして扱われます.

〔例1〕

A%	整数変数
PNT!	単精度変数
MAX#	倍精度変数
L$	文字変数
AB	単精度変数

〔例2〕

VARIABLE%
VARIABLE!
VARIABLE#
VARIABLE$

これらは異なる変数名として区別されますが，VARIABLE と VARIABLE！は同じ変数になります.

2.5.2 配列変数

配列は，同じ性質を持つ複数個のデータの集まりです．配列は同一名でその要素を参照することができ，それぞれの要素は添字により順序付けられます.

ある変数を配列変数として宣言するためには．DIM 文を用いて次のように行います.

DIM 変数名（添字の最大値 [，添字の最大値]……）

カッコの中に書かれた添字の最大値の個数により，その配列変数の次元数を指定します．配列の次元は，1次元から多次元の指定が行えます．添字の最大値は整数形式で，0から32766の範囲でなければなりません．但し，メモリにより配列の多次元指定および添字の最大値は制限されます．また各次元の添字の最大値が10を越えなければ，DIM 文なしに配列変数を使うことができます.

配列変数の一要素の参照は，各次元の添字を指定した添字付き名で行い，その形式は次のとおりです.

変数名（第1次元の添字 [，第2次元の添字]……）

変数名の直後にカッコでくくって各次元の添字を指定します．この中に書かれた添字の個数は，配列変数の次元数と一致しなければなりません．添字は数値式で，その値は0から添字の最大値までです．各次元の添字の結果が整数でない場合は，小数部を四捨五入した整数に変換されます.

〔例1〕 1次元の配列

DIM A$(5)

A$(0)
A$(1)
A$(2)
A$(3)
A$(4)
A$(5)

〔例2〕 2次元の配列

配列名Bは次のように行と列のテーブルとして考えられます.

DIM B(2,3)

列

行				
	B (0,0)	B (0,1)	B (0,2)	B (0,3)
	B (1,0)	B (1,1)	B (1,2)	B (1,3)
	B (2,0)	B (2,1)	B (2,2)	B (2,3)

2.5.3 キーワード

キーワードは, BASICで予めその意味および用途が定められた言葉です. キーワードは, すべて予約語であり, 変数名としての使用または変数名をキーワードで始めることはできません.

F-BASICのキーワードには, コマンド名, ステートメント名, 関数名, および演算子があります.

キーワードの省略形 以下に示すキーワードには, 省略形が許されています. プログラムをキーワードの完全形, 省略形のいずれで入力してもプログラムリストは完全形で出力されます.

キーワード	省略形
CONT	C.
LOAD	LO.
SAVE	SA.
SKIPF	SK.
MERGE	ME.
GOTO	GO.
GOSUB	GOS.
RETURN	RET.
RANDOMIZE	RNDM.
LOCATE	LOC.
COLOR	COL.
MOTOR	M.
RUN	R.
LIST	L.
CONSOLE	CONS.
WIDTH	W.

2.5.4 予約語

○印…予約語，×印…予約語でない

予約語	V1.0		V2.0	V3.0	
	R	D		R	D
ABS	○	○	○	○	○
AND	○	○	○	○	○
ANPORT	○	○	○	○	○
ASC	○	○	○	○	○
ATN	○	○	○	○	○
AUTO	○	○	○	○	○
BEEP	○	○	○	○	○
BUBINI	○	○	○	○	○
BUBR	○	○	○	○	○
BUBW	○	○	○	○	○
CDBL	○	○	○	○	○
CHAIN	×	×	○	○	○
CHR$	○	○	○	○	○
CINT	○	○	○	○	○
CIRCLE	○	○	○	○	○
CLEAR	○	○	○	○	○
CLOSE	○	○	○	○	○
CLS	○	○	○	○	○
COLOR	○	○	○	○	○
COM	○	○	○	○	○
COMMON	×	×	○	○	○
CONNECT	○	○	○	○	○
CONSOLE	○	○	○	○	○
CONT	○	○	○	○	○
COS	○	○	○	○	○
CSNG	○	○	○	○	○
CSRLIN	○	○	○	○	○
CVD	×	○	○	×	○
CVI	×	○	○	×	○
CVS	×	○	○	×	○
DATA	○	○	○	○	○
DATE	○	○	○	○	○
DEF	○	○	○	○	○
DEFDBL	○	○	○	○	○
DEFINT	○	○	○	○	○

予約語	V1.0		V2.0	V3.0	
	R	D		R	D
DEFSNG	○	○	○	○	○
DEFSTR	○	○	○	○	○
DELETE	○	○	○	○	○
DIM	○	○	○	○	○
DSKF	×	○	○	×	○
DSKI$	×	○	○	×	○
DSKINI	×	○	○	×	○
DSKO$	×	○	○	×	○
EDIT	○	○	○	○	○
ELSE	○	○	○	○	○
END	○	○	○	○	○
EOF	○	○	○	○	○
EQV	○	○	○	○	○
ERASE	×	×	○	○	○
ERL	○	○	○	○	○
ERR	○	○	○	○	○
ERROR	○	○	○	○	○
EXEC	○	○	○	○	○
EXP	○	○	○	○	○
FIELD	×	○	○	×	○
FILES	○	○	○	○	○
FIX	○	○	○	○	○
FN	○	○	○	○	○
FOR	○	○	○	○	○
FRE	○	○	○	○	○
GCURSOR	○	○	○	○	○
GET	○	○	○	○	○
GO	○	○	○	○	○
HARDC	○	○	○	○	○
HEX$	○	○	○	○	○
IF	○	○	○	○	○
IMP	○	○	○	○	○
INKEY$	○	○	○	○	○

予約語	V1.0		V2.0	V3.0	
	R	D		R	D
INPUT	○	○	○	○	○
INSTR	○	○	○	○	○
INT	○	○	○	○	○
INTERVAL	○	○	○	○	○
KEY	○	○	○	○	○
KILL	○	○	○	○	○
LEFT$	○	○	○	○	○
LEN	○	○	○	○	○
LET	○	○	○	○	○
LINE	○	○	○	○	○
LIST	○	○	○	○	○
LLIST	×	×	○	○	○
LOAD	○	○	○	○	○
LOC	×	○	○	×	○
LOCATE	○	○	○	○	○
LOF	○	○	○	○	○
LOG	○	○	○	○	○
LPOS	×	×	○	○	○
LPRINT	×	×	○	○	○
LSET	×	○	○	×	○
MERGE	○	○	○	○	○
MID$	○	○	○	○	○
MKD$	×	○	○	×	○
MKI$	×	○	○	×	○
MKS$	×	○	○	×	○
MOD	○	○	○	○	○
MON	○	○	○	○	○
MOTOR	○	○	○	○	○
NAME	×	○	○	×	○
NEW	○	○	○	○	○
NEXT	○	○	○	○	○
NOT	○	○	○	○	○
OCT$	○	○	○	○	○
OFF	○	○	○	○	○
ON	○	○	○	○	○
OPEN	○	○	○	○	○

予約語	V1.0		V2.0	V3.0	
	R	D		R	D
OR	○	○	○	○	○
PAINT	○	○	○	○	○
PEEK	○	○	○	○	○
PEN	○	○	○	○	○
PLAY	×	×	×	○	○
POINT	○	○	○	○	○
POKE	○	○	○	○	○
POS	○	○	○	○	○
PRESET	○	○	○	○	○
PRINT	○	○	○	○	○
PSET	○	○	○	○	○
PUT	○	○	○	○	○
RANDOMIZE	○	○	○	○	○
READ	○	○	○	○	○
REM	○	○	○	○	○
RENUM	○	○	○	○	○
RESTORE	○	○	○	○	○
RESUME	○	○	○	○	○
RETURN	○	○	○	○	○
RIGHT$	○	○	○	○	○
RND	○	○	○	○	○
RSET	○	○	○	○	○
RUN	○	○	○	○	○
SAVE	○	○	○	○	○
SCREEN	○	○	○	○	○
SGN	○	○	○	○	○
SIN	○	○	○	○	○
SKIPF	○	○	○	○	○
SOUND	×	×	×	○	○
SPACE$	○	○	○	○	○
SPC	○	○	○	○	○
SQR	○	○	○	○	○
STEP	○	○	○	○	○
STOP	○	○	○	○	○
STR$	○	○	○	○	○
STRING$	○	○	○	○	○
SUB	○	○	○	○	○
SWAP	○	○	○	○	○

予約語	V1.0		V2.0	V3.0		予約語	V1.0		V2.0	V3.0	
	R	D		R	D		R	D		R	D
SYMBOL	○	○	○	○	○	USING	○	○	○	○	○
						USR	○	○	○	○	○
TAB(	○	○	○	○	○						
TAN	○	○	○	○	○	VAL	○	○	○	○	○
TERM	○	○	○	○	○	VARPTR	○	○	○	○	○
THEN	○	○	○	○	○						
TIME	○	○	○	○	○	WEND	○	○	○	○	○
TO	○	○	○	○	○	WHILE	○	○	○	○	○
TROFF	○	○	○	○	○	WIDTH	○	○	○	○	○
TRON	○	○	○	○	○						
						XOR	○	○	○	○	○
UNLIST	○	○	○	○	○						

R…ROMモード、D…DISKモード

2.6 型変換

BASICのステートメントに書かれた数値データは，実行のとき必要に応じて，その型から他の型に変換されます．以下にその変換の規則について述べます．

（1） ある型の数値データが，それと異なる型の数値変数に代入されるとき，数値データは，その変数名に宣言された型に変換して代入されます．

〔例〕
```
10 A%=12.34
20 PRINT A%
RUN
12
Ready
```

（2） ある型の数値データが，それよりも精度の低い変数に代入されるとき，四捨五入が行われます．たとえば，倍精度の数値データを単精度変数に代入するような場合です．

〔例〕
```
10 A=12.3456789#
20 PRINT A
RUN
 12.3457

Ready
```

実数データを整数変数に代入するときは，小数点以下が四捨五入され，整数値に変換されます．このとき，変換されたデータが整数型で表わせる範囲をこえているときは，エラーになります．

〔例1〕
```
10 A%=12.34
20 B%=35.56
30 PRINT A%, B%
RUN
 12              36

Ready
```

〔例2〕
```
10 A%=1.23E+06
20 PRINT A%
RUN
Overflow In 10

Ready
```

以上のような実数値から整数値への変換は，代入文のみならず，関数及びステートメントの評価のときにも行なわれます．具体的には，以下のような場合です．

・配列要素の添字が実数のとき
・キヤラクタ及びグラフィック座標が実数のとき
・ファイル番号が実数のとき
・プログラマブルファンクションキーの番号が実数のとき
・オペランドに整数値を要求するステートメントで，実数が用いられたとき
・引数に整数値を要求する関数の引用において，実数が用いられたとき

(3) 精度の低い数値データをより精度の高い数値変数に代入するときは，その変数の型に変換後代入されます．しかし，結果として得られる高精度の数値は，もとのデータよりも精度が高くなることはありません．たとえば，Bという単精度のデータをA#という倍精度変数に代入したとすると，A#の最初の6桁のみ正しい精度が得られます．これは，Bというデータが6桁の精度しか持っていないからです．

この変換における相対誤差は，次の式で表わすことができます．

相対誤差=ABS((A#−B)/B)<5.96E−08

すなわち，倍精度の数値ともとの単精度の数値との差をもとの単精度の数値で割った値の絶対値は，5.96E−08より小さくなります．

〔例〕
```
10 B=3.02
20 A#=B
30 PRINT B, A#
RUN
 3.02        3.019999980926514

Ready
```

(4) 精度の異なる数値の間での算術演算及び関係演算の場合には，低い精度を高い精度に変換後，同じ精度で演算が行われます．したがって，演算の結果は高い方の精度になります．

〔例1〕

```
10 A#=7#/6
20 B#=7#/6#
30 PRINT A#, B#
RUN
 1.1666666666666667
 1.1666666666666667

Ready
```

7#／6の除算は倍精度で行われ，その結果が倍精度でA#に返されます．

〔例2〕

```
10 A=7#/6
20 PRINT A
RUN
 1.16667

Ready
```

7#／6の除算は倍精度で行われますが，Aは単精度変数であるため，その結果が単精度に丸められAに代入されます．

(5) 論理演算の場合，扱う数値はすべて整数に変換され演算が行われます．その結果も整数値で返されます．整数値に変換したとき，その値は－32768から32767の範囲になければなりません．そうでなければ，"Overflow" のエラーになります．

〔例1〕

```
10 A#=34.56
20 B=NOT A#
30 PRINT B, A#
RUN
-36           34.5600013732910 2

Ready
```

〔例2〕

```
10 A#=34.56D7
20 B=NOT A#
30 PRINT B, A#
RUN

Overflaw In 20
Ready
```

2.7 式

式は演算の文法を示すものであり，定数や変数，関数を演算子で結んで表わします．式の演算結果は，一個の数値又は一個の文字列になりますから，単に数値や文字，あるいは変数だけのものも式と呼びます．式を分類すると，次の4つになります．

・算 術 式
・関 係 式
・論 理 式
・文 字 式

これらの中で，算術式，関係式及び論理式は，その評価結果が数値であることから，まとめて数値式と呼びます．

2.7.1 算 術 式

算術式は，変数，定数，関数などの数値データを算術演算子で結んだものです．また，算術演算子のない数値データのみの場合も算術式と呼びます．

算術演算子	内 容	表 記
＋	加算を行います．	A＋B
－	減算を行います．	A－B
＊	乗算を行います．	A＊B
／	浮動小数点除算を行います．	A／B
^	ベキ乗を行います．	A^ B
¥	整数の除算を行います． 結果は，浮動小数点除算の小数点以下を切り捨てた値となります．	A¥B
MOD	整数の除算の剰余を求めます．	A MOD B

整数の除算（¥）および，整数の剰余（MOD）の演算において，扱う数値が整数形式でないときは，小数部分を四捨五入し，整数に変換してから演算が行われます．

〔例〕　24．35¥6．87＝24¥7＝3
　　　10．2MOD4＝10MOD4＝2

2.7.2 関　係　式

関係式は2つの値を比較するときに用い，比較の結果，真ならー1が偽ならば0が設定されます．

関係演算子	内　容	表　記
=	等しい	A=B
<>, ><	等しくない	A<>B
<	小さい	A<B
>	大きい	A>B
<=, =<	等しいか小さい	A<=B
>=, =>	等しいか大きい	A>=B

(1) 数値の比較

数値の比較はIF文により，プログラムの流れを変えることができます．

〔例1〕

```
10 A=10:B=20
20 IF A>B THEN PRINT "A>B"
          ELSE PRINT "A<B"
RUN
 A<B

Ready
```

〔例2〕

```
10 A=10<20:B=10>20
20 PRINT A;B
RUN
 -1   0

Ready
```

(2) 文字の比較

文字列の比較は，各文字の一字一字に対応するキャラクターコード（付録キャラクタコード表参照）で比較します．すべてのキャラクタコードが等しければ，文字列は等しくなります．キャラクタコードが等しくなれば，小さいキャラクタコードの文字列は大きいキャラクタコードの文字列よりも小さくなります．もし比較の途中で，文字列が終りになったら，短い文字列が小さくなります．文字列のなかの空白も比較の対象になりますので注意してください．

〔例〕

```
"ABC" = "ABC"
"AAB" < "AAC"
"A&" > "A#"
"PR  " > "PR"
```

〝cm〟 > 〝CM〟
〝LOAD〟 < 〝LOADM〟
A$< 〝02：37：15〟（A$＝〝01：25：48〟のとき）

2.7.3 論 理 式

ビット操作や論理演算を行なったり，いくつかの関係式を調べたりするときに，論理式を用います．論理式は，論理演算子と変数，定数，関数などの数値データの結合により構成されます．なお，数値データが整数でないときは，小数点以下を四捨五入した整数に変換されます．

論理演算子にはNOT，AND，OR，XOR，IMP，EQVがあり，演算は対応するビット単位に行われます．

論理演算の結果を次に示します．

（1） NOT（否定）

X	NOT X
1	0
0	1

（2） AND（論理積）

X	Y	X AND Y
1	1	1
1	0	0
0	1	0
0	0	0

（3） OR（論理和）

X	Y	X OR Y
1	1	1
1	0	1
0	1	1
0	0	0

（4） XOR（排他的論理和）

X	Y	X XOR Y
1	1	0
1	0	1
0	1	1
0	0	0

（5） IMP（包含）

X	Y	X IMP Y
1	1	1
1	0	0
0	1	1
0	0	1

（６） ＥＱＶ（同値）

X	Y	X EQV Y
1	1	1
1	0	0
0	1	0
0	0	1

〔例〕

（１） ＮＯＴ（否定）

Ａ％＝ＮＯＴ　１

$1=(0000000000000001)_2$ によりＮＯＴ　$1=(1111111111111110)_2=-2$

（２） ＡＮＤ（論理積）

Ｂ％＝２　ＡＮＤ　３

$2=(0000000000000010)_2$,　$3=(0000000000000011)_2$ より

B％$=(0000000000000010)_2=2$

（３） ＯＲ（論理和）

Ｃ％＝－１　ＯＲ　－４

$-1=(1111111111111111)_2$,　$-4=(1111111111111100)_2$ より

C％$=(1111111111111111)_2=-1$

（４） ＸＯＲ（排他的論理和）

Ｄ％＝４ ＸＯＲ　－３

$4=(0000000000000100)_2$,　$-3=(1111111111111101)_2$ より

D％$=(1111111111111001)_2=-7$

（５） ＩＭＰ（包含）

Ｅ％＝１　ＩＭＰ　３

$1=(0000000000000001)_2$,　$3=(0000000000000011)_2$ より

E％$=(1111111111111111)=-1$

（６） ＥＱＶ（同値）

Ｆ％＝５　ＥＱＶ　－３

$5=(0000000000000101)_2$,　$-3=(1111111111111101)_2$ より

F％$=(0000000000000111)_2=7$

2.7.4 文字式

文字変数または，文字定数を演算子プラス（＋）によって連結した式を文字式と呼びます．文字式で用いられる演算子のプラス（＋）は，2つの文字データの加算ではなく，2つの文字列を連結することを意味します．また単に1つの文字変数または，文字定数をも文字式と呼びます．

〔例〕

```
10  A$="FILE"
20  B$="NAME"
30  PRINT  A$+B$+"=";
RUN
FILENAME=
```

2.7.5 演算子の優先順位

式の中の演算は，優先順位の高いものから実行され，同じ優先順位の演算子のときは，左の方から先に実行されます．

演算子の優先順位は次のようになります．

優先順位	演算子
1	^
2	－（負符号）
3	＊，／
4	¥
5	MOD
6	＋，－
7	関係演算子
8	NOT
9	AND
10	OR
11	XOR
12	EQV
13	IMP

カッコで囲まれた式および関数の引用は，演算子の優先順位に関係なく先に行われます．

〔例〕

```
A＊B＋C＊D
①└─┘  └─┘②
  └──③──┘

(A＋B)＊SIN((X＋Y)／Z)
①└─┘        ②└─┘ ③
  │      ④└────┴──┘
  └───⑤────┘
```

2.8 ファイルディスクリプタ

F-BASICでは，全ての入出力装置をファイルという概念で扱います．ファイルとは，入出力媒体上の意味を持った論理的なデータの集合のことで，F-BASICには2つのファイル形式があります．1つはプログラムファイルであり，もう1つはデータファイルです．ファイルは，ファイルディスクリプタにより区別されます．

(1) ファイル番号

F-BASICでの入出力操作はファイル番号を通して行われます．ファイル番号は，ファイルディスクリプタに対して割当てられた論理番号で，OPEN文により定義されます．

ファイル番号は1から16の範囲であり，定数，変数または式で表します．

(2) ファイルディスクリプタの形式

ファイルディスクリプタは，次の形式の文字列で構成されます．

"[デバイス名][[(オプション)][ファイル名]]"

デバイス名，オプション，ファイル名は省略してもよい場合があります．ファイルディススリプタは通常，上記のように全体を引用符で囲って記述されますが，文字変数または文字式の文字表記であってもかまいません．

(3) デバイス名

F-BASICで使用できる入出力装置のデバイス名は，次のとおりです．

入出力装置名		デバイス名	入力	出力	備考
キーボード		KYBD:	○	×	
スクリーン		SCRN:	×	○	
プリンタ		LPT0:	×	○	
RS-232Cポート	0	COM0:	○	○	ポート1からポート4は
	1	COM1:	○	○	拡張ユニットのRS-232C
	2	COM2:	○	○	ポートです．
	3	COM3:	○	○	
	4	COM4:	○	○	
カセットテープ		CAS0:	○	○	ROMモードでの省略値
フロッピィディスク	0	0:	○	○	DISKモードでの省略値
	1	1:	○	○	
	2	2:	○	○	
	3	3:	○	○	
バブルカセット	0	BUB0:	○	○	
	1	BUB1:	○	○	

(4) オプション

オプションは，6文字以内の英数字をカッコで囲って指定し，RS-232C通信機能における，モードの指定を行います．

(5) ファイル名

ファイル名は次の規則に従った文字列です．

(1) 8文字以内でなければなりません．

(2) 文字列の中にコロン(：)，引用符(〃)，左カッコおよび右カッコを含んではなりません．
また，バブルカセットおよびディスクのとき，空ファイル名であってはなりません．

ファイル名の前に書かれたデバイス名は，そのファイルが存在する機器を示します．このデバイス名が省略されたときは，ROMモードのときは〝CAS0：〃が，DISKモードのときは〝0：〃が指定されたものと見なされます．

(6) ファイルディスクリプタの記述例

〝CAS0：FILE1〃	カセットテープ
〝FILE1〃	カセットテープ，ROMモードでの省略値
〝0：FILE2〃	フロッピィディスク0
〝FILE2〃	フロッピィディスク0，DISKモードでの省略値
〝BUB0：FILE3〃	バブルカセット0
〝COM0：(S8N2)〃	RS-232Cポート0

2.9 初　期　設　定

電源投入またはリセットボタンの押下時のディスプレイの画面，RS-232Cポート，プログラマブルファンクションキー、文字領域は、次の状態に設定されています．

(1) ディスプレイ画面

1画面の表示	40字×20行
スクロール開始行	0行
スクロール行数	20行
プログラマブルファンクションキーの表示	表示なし
表示色	白

これらはWIDTH文，CONSOLE文で変更可能です．

(2) RS-232Cポート

クロック	Slowクロック（1／64分周）
ビット長	8ビット
パリティ	パリティなし
ストップビット	2ビット
モード	全二重通信
オートLF	オートLFしない

これらの状態はTERM文で変更可能です．

（3） プログラマブルファンクションキー

各プログラマブルファンクションキーには，次の文字列が設定されています．

・F-BASIC　V1.0，V2.0の場合

ROMモード，DISKモード共に次の文字列が設定されています．

PF1　AUTO␣
PF2　LIST ®
PF3　RUN ®
PF4　GO␣TO␣
PF5　CONT ®
PF6　?DATE$，TIME$ ®
PF7　LIST"LPT0:" ®
PF8　KEY␣
PF9　LOAD␣
PF10　HARDC ®

␣はスペースを表し，®はプログラマブルファンションキーが押された時点で，すぐに文字列の命令を実行することを意味します．

・F-BASIC　V3.0の場合

ROMモード，DISKモードにより初期設定は次のようになります．

PFキー ＼ モード	ROMモード	DISKモード
PF1	AUTO␣	AUTO␣
PF2	LIST ®	LIST ®
PF3	RUN ®	RUN ®
PF4	CONT ®	CONT ®
PF5	LLIST ®	LLIST ®
PF6	LOAD ®	LOAD"
PF7	SAVE"	SAVE"
PF8	?DATE$，TIME$ ®	FILES
PF9	SCREEN 7，7 ®	SCREEN 7，7 ®
PF10	HARDC ®	HARDC ®

␣はスペースを表し，® はプログラマブルファンクションキーが押された時点で，すぐに文字列の命令を実行することを意味します．

（4） 文字領域

文字式などで使用する領域で，300バイトに設定されています．この値はCLEAR文により変更することができます．

2.10 ファイルの構成と管理

F-BASICでは，データの入出力をファイルという概念により取扱っています．ファイルとは，入出力媒体上の意味を持ったデータの集合をいい，使用者が定義したファイル名で，OPENしてから，CLOSEするまでの間に出力されたデータが，一つのファイルになります．

ここでは，データレコーダ，バブルカセット，ミニフロッピィディスク，標準フロッピィディスクの構成と管理について説明します．ただし，バブルカセットはV1.0，V2.0で適用されます．

2.10.1 データレコーダ

F-BASICでは次の形式で記録されます．

ブロックタイプ

0 0……ヘッダブロック
0 1……データブロック
F F……エンドブロック

ブロック長さ

データの長さ（バイト数）を示します（00～FF）．

データ

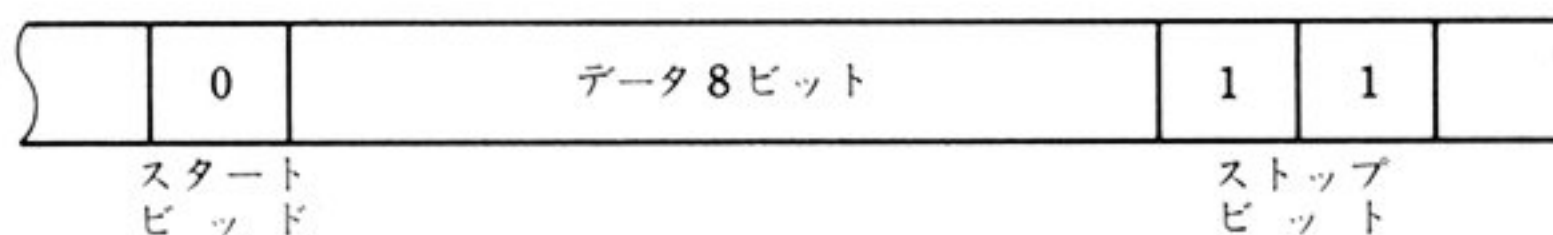
0～255バイトまでのデータが入っています．

ヘッダブロックのときは，ファイル名，ファイル形式，属性（アスキー／バイナリ）が記録され，エンドブロックのときは，データは何も記録されません．

チェックサム

ブロックタイプ，ブロック長さ，データに対するチェックコード．

なお，1バイトのデータは次の形式になっています．

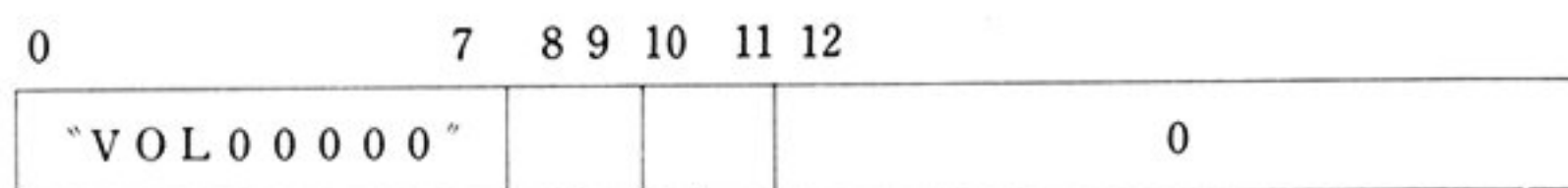
	0	データ8ビット	1	1	
	スタート ビット		ストップ ビット		

2.10.2 バブルカセット

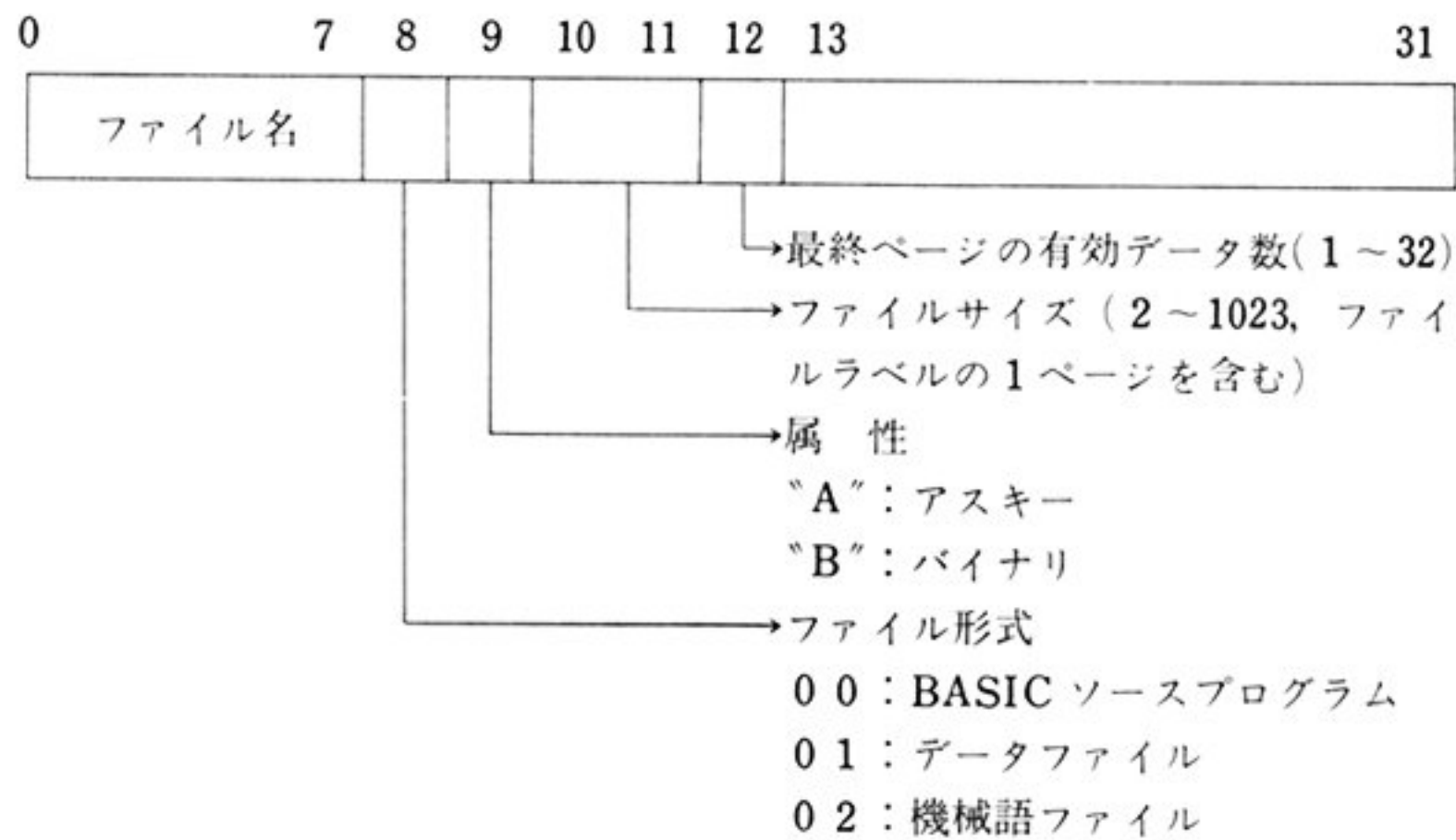
バブルカセットは，32バイトを1ページとするページアドレスが割当てられています．32KBバブルカセットのアドレス割当てを次に示します．

0	1	2	3	4	5	6	7
8	9	10	11	12	13	14	15
1008	1009	1010	1011	1012	1013	1014	1015
1016	1017	1018	1019	1020	1021	1022	1023

F-BASICでは，このページアドレスを指定して，直接データの書込み読出しが行えます．ファイルとして取扱うときは，ページアドレスを考えることなくバブルカットを使用できます．このときF-BASICは，下記に示すボリュームラベル，ファイルラベルによりバブルカセットを管理しています．

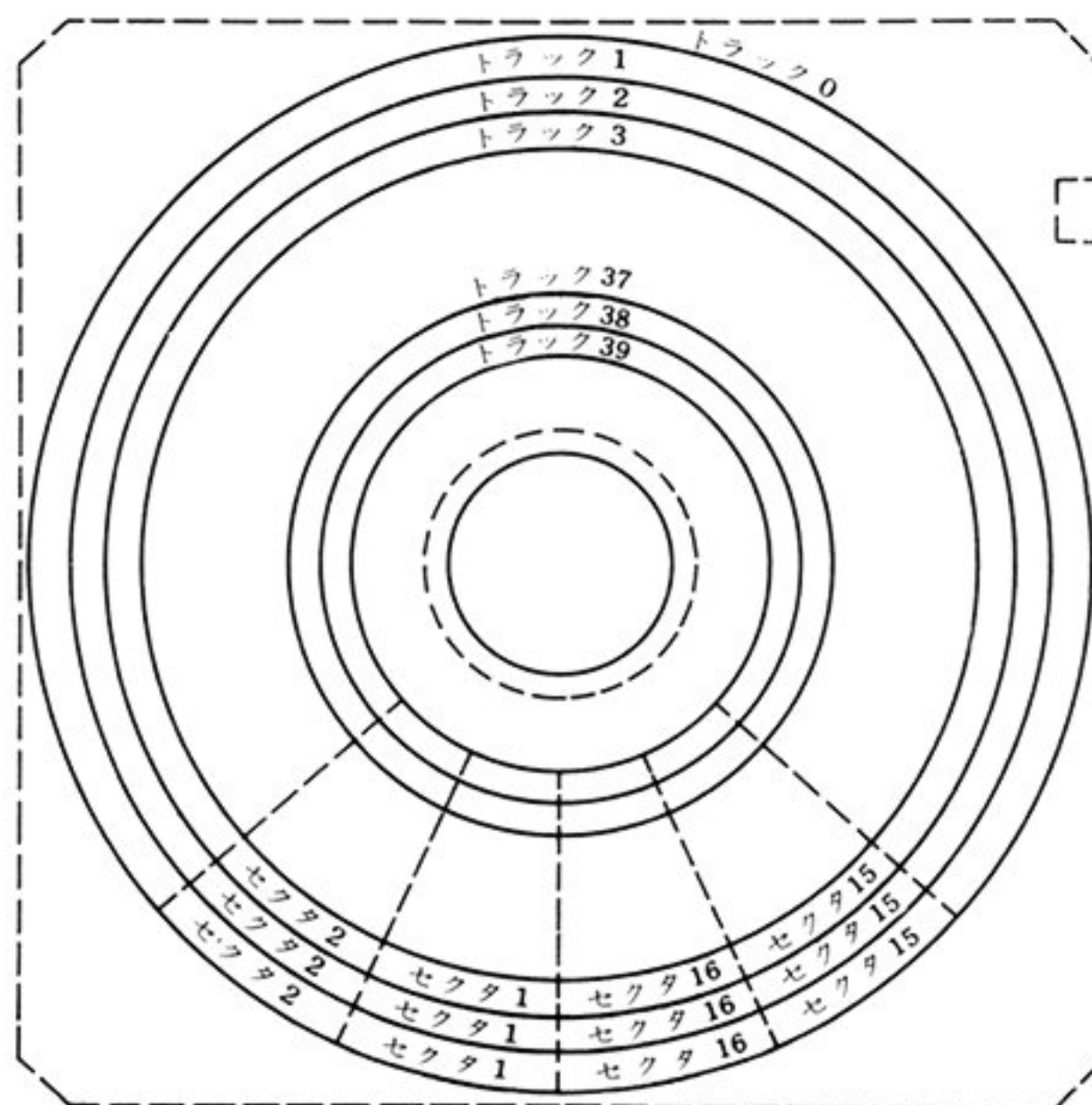
ボリュームラベル

第0ページに付けられます．

ファイルラベル

ファイルの先頭ページに付けられます.

バブルカセットを初めて使用するときは, BUBINIコマンドにより初期化する必要があります.

2.10.3 ミニフロッピィディスク

〔構　造〕

F-BASICで使用するミニフロッピィディスクは, 5.25インチ両面倍密度型で, その構造は次のようになっています.

ミニフロッピィディスクの表をサイド0，裏をサイド1と呼び，F-BASICにおけるディスク1枚の主な仕様は，次のようになっています．

記憶容量	320Kバイト
シリンダ数	40シリンダ
トラック数	80トラック（40トラック×2）
セクタ数	16セクタ／トラック
セクタ長	256バイト／セクタ

〔セクタアドレス〕

ミニフロッピィディスクのある特定のセクタを指定するには，トラック番号とセクタ番号を用いて行います．両面タイプの場合は，さらにサイド番号が必要ですが，F-BASICでは，サイド0，サイド1の全セクタに通し番号を付け，その通し番号とトラック番号により，セクタを指定します．物理的なセクタ番号との対応を次に示します．

	サイド0			サイド1		
セクタ	1		16	1		16
トラック　0	セクタ1	--------	セクタ16	セクタ17	--------	セクタ32
⋮						
トラック　39	セクタ1	--------	セクタ16	セクタ17	--------	セクタ32

〔クラスタ〕

ミニフロッピィディスクの読み書きの最小単位は1セクタですが，F-BASICでは，8セクタを管理の最小単位として扱います．この単位をクラスタと呼びます．

トラック2のセクタ1～8がクラスタ0，トラック39のセクタ25～32がクラスタ151となります．

〔トラックの割当て〕

F-BASICで使用するミニフロッピィディスクの各トラックは，次の表のように割当てられています．

トラック番号	内　　　容
0	IPL，ID，DISKコード
1	FAT，ディレクトリ
2 ～ 39	ユーザ領域

・IPL………Diskコードをメモリ上に展開するプログラムです．
（アイ・ピー・エル：Initial　Program　Loader）

・ID………F-BASICで使用できるディスクであるかを識別する領域です．
（アイ・ディー：Identification）

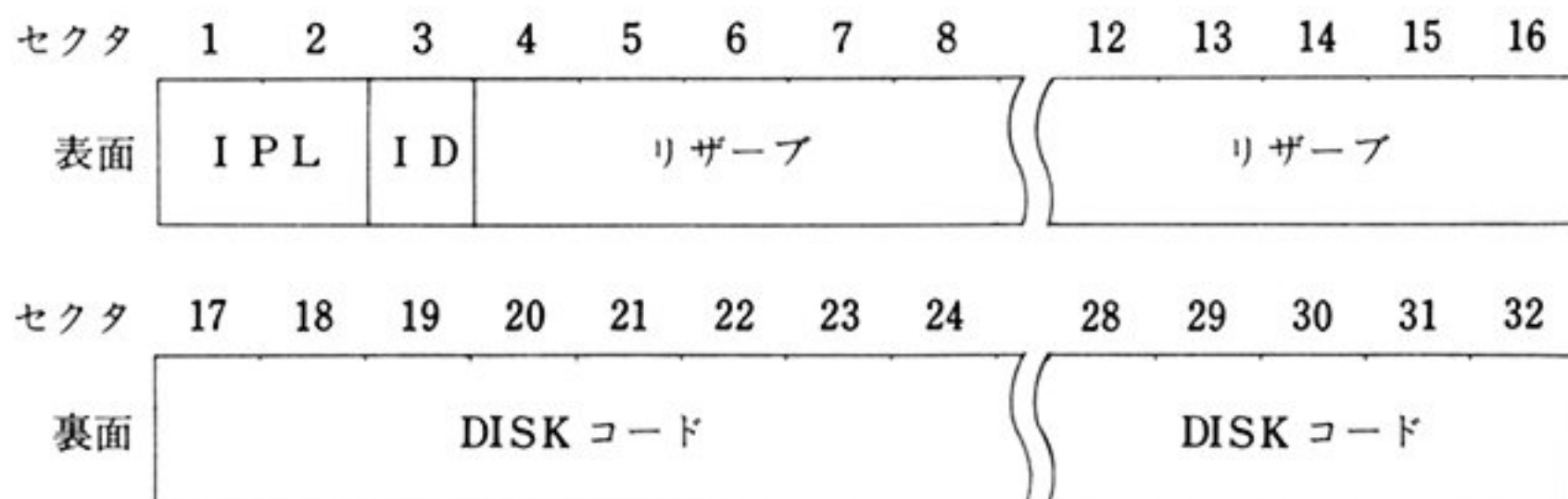
・DISKコード………ディスクに関する命令が格納されています．この領域は，DISKモード起動時にメモリの&H7000番地以降に展開されます．

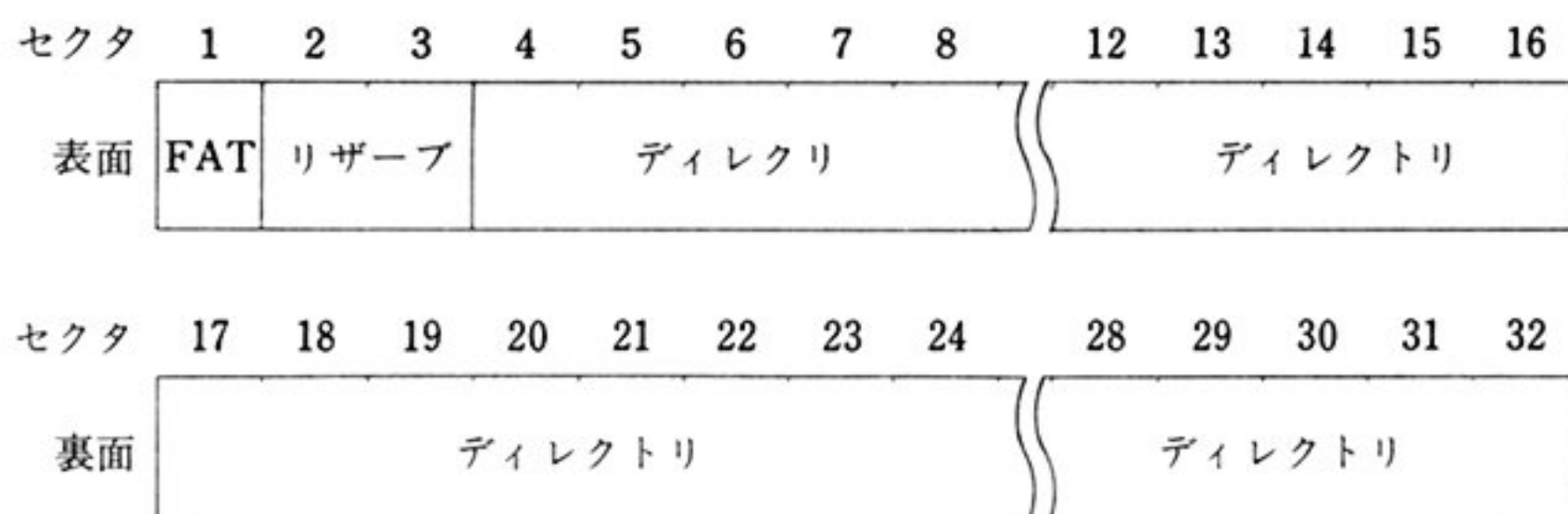
・FAT………各クラスタの使用状況を記録する領域です．
（ファット：File　Allocation　Table）

・ディレクトリ………ファイル名，ファイル形式，ファイル位置を記録している領域です．

・ユーザ領域………ユーザが実際にファイルとして使用可能な領域です．

〔トラック0，1の構成〕

トラック0

トラック1

〔FAT〕

FATは各クラスタの使用状況を示すテーブルです．F-BASICはディスクの領域の管理を，このFATによりすべて行っています．FATはトラック1の最初のセクタ内に書込まれ，6～157バイトがクラスタの0～151に対応して，各バイトは次の意味を持っています．

値 (16進)	意　味（クラスタの使用状況）
0～97	このクラスタは使用中であり，後続するクラスタを持つ．又，その値が後続するクラスタの番号を示す．
C0～C7	このクラスタは使用中であり，ファイル最後のクラスタである．この値から16進のBFを引いた値は，クラスタ内の実際に使用されているセクタ数を示す．
FD	クラスタ中の使用セクタはない．
FE	このクラスタはシステム領域である．
FF	このクラスタは未使用である．

〔ディレクトリ〕

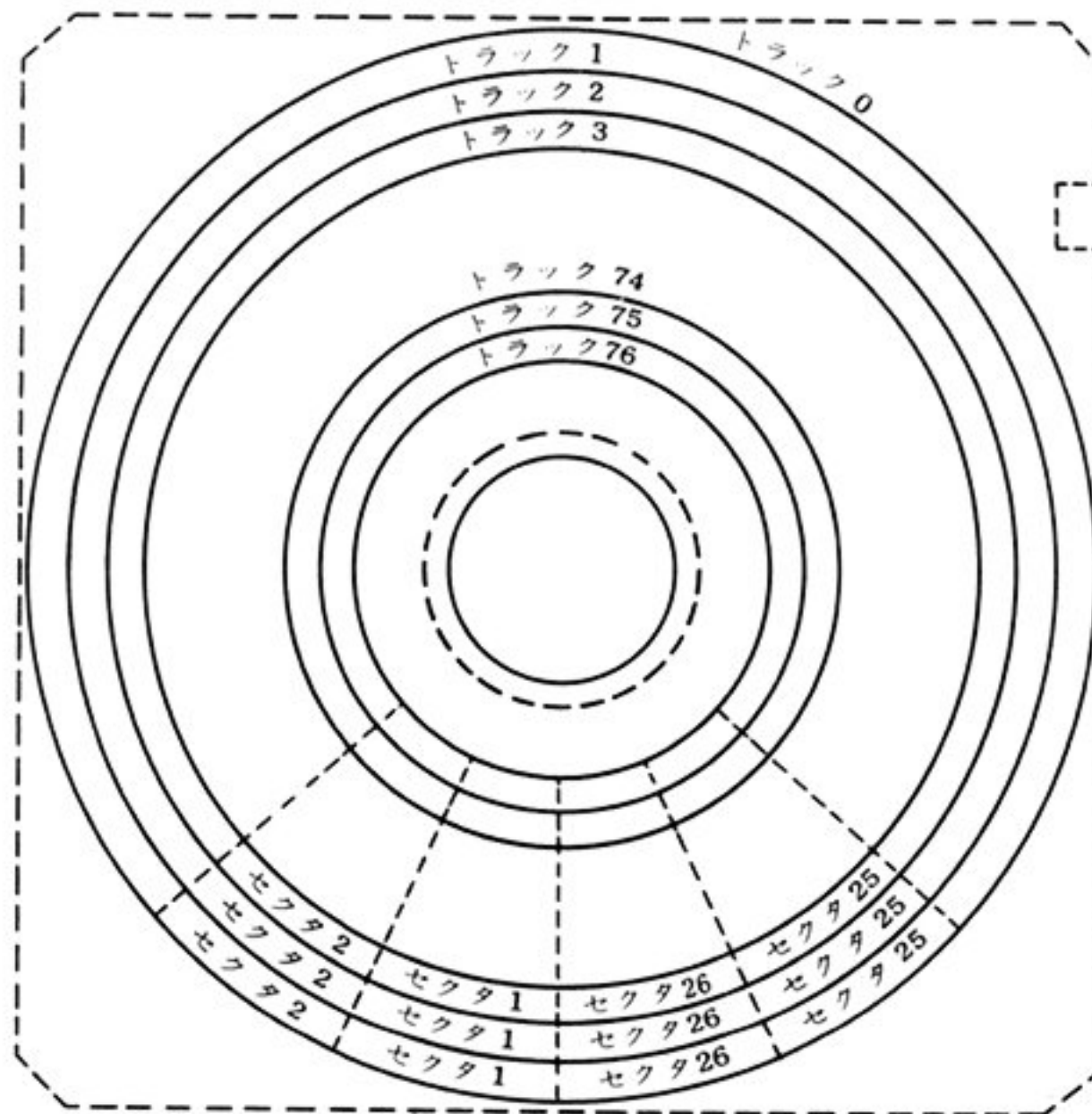
ディレクトリは，トラック1のセクタ4～セクタ32に書込まれます．ディレクトリ部のセクタは，32バイトずつの8ブロックに分割されます．この32バイトは，ディレクトリスロットと呼ばれ，次の構成になっています．

ディレクトリスロット内バイト位置	内　　容
0～7	ファイル名
8～10	リザーブ用
11	ファイルタイプ 00：BASICソース 01：BASICデータ 02：機械語ファイル
12	アスキーフラグ FF：アスキー 00：バイナリ
13	ランダムアクセスフラグ
14	ファイル先頭クラスタ番号
15～31	リザーブ用

ディレクトリスロットは232個ありますので，最大232のファイルが登録可能ですが，クラスタ数は152なので，実際登録できるファイル数は最大152となります．

2.10.4　標準フロッピィディスク

〔構　造〕

F-BASIC使用する標準フロッピィディスクは，8インチ両面倍密度型で，その構造は次のようになっています．

標準フロッピィディスクの表をサイド0，裏をサイド1と呼ぶ．F-BASICにおけるディスク1枚の主な仕様は，次のようになっています．

記憶容量	1MB
シリンダ数	77シリンダ
トラック数	154トラック（77シリンダ×2）
セクタ数	26セクタ／トラック
セクタ長	256バイト／セクタ

〔**セクタアドレス**〕

標準フロッピィスィスクのある特定のセクタを指定するには，トラック番号とセクタ番号を用いて行います．両面タイプの場合は，さらにサイド番号が必要ですが，F-BASIC, V2.0はサイド0，サイド1の全セクタに通し番号をつけ，その通し番号とトラック番号によりセクタを指定します．

物理的なセクタとの対応を次に示します．

	サイド0			サイド1		
セクタ	1		26	1		26
トラック 0	セクタ1	…………	セクタ26	セクタ27	…………	セクタ52
⋮						
トラック 76	セクタ1	…………	セクタ26	セクタ27	…………	セクタ52

〔クラスタ〕

F-BASICがファイルを管理するための単位で，26セクタを1クラスタとしています．

1（クラスタ）=26（セクタ）×256（バイト）　→6,656（バイト）

クラスタとトラック，セクタの関係を次の表に示します．

クラスタ番号	トラック	セクタ
0	1（表面）	1～26
1	1（裏面）	27～52
2	2（表面）	1～26
3	2（裏面）	27～52
150	76（表面）	1～26
151	76（裏面）	27～52

〔トラックの割当て〕

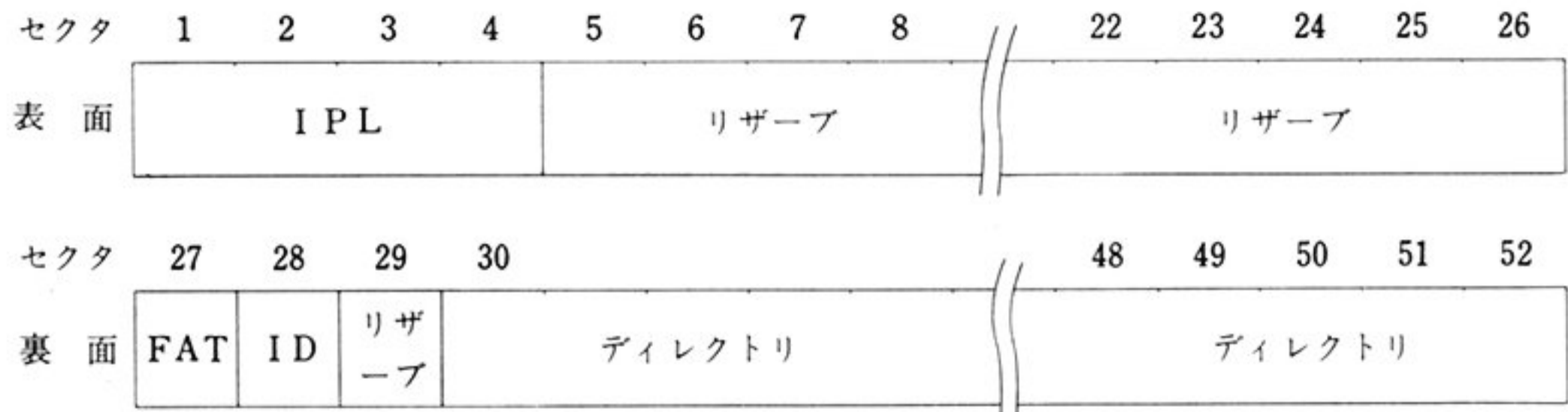
F-BASICで使用する標準フロッピィディスクの各トラックは，次の表のように割当てられています．

トラック番号	システム・ディスク	ユーザ・ディスク
0	IPL FAT，ID，ディレクトリ	IPL FAT，ID，ディレクトリ
1 ～ 3	システム・コード	ユーザ領域
4 ～ 76	ユーザ領域	ユーザ領域

・システム・ディスク………F-BASIC　V3.0のシステムコードを含むディスク
・ユーザ・ディスク…………システムコードを含まないディスク
・IPL……システムコードをメモリ上に展開するプログラムです．
　（アイ・ピー・エル：Initial Program Loader）
・FAT……各クラスタの使用状況を記録する領域です．
　（ファット：File Allocation Table）
・ID……システムディスクかユーザディスクであるかを識別する領域です．
　（アイ・ディー：Identification）
・ディレクトリ……ファイル名，ファイル形式，ファイル位置を記録している領域です．
・システムコード……F-BASIC　V3.0のインタプリタです．
・ユーザ領域……ユーザが実際にファイルとして使用可能な領域です．

〔トラック0の構成〕

次の図のようにIPL, FAT, ID, ディレクトリがトラック0の特定セクタに格納されています.

〔FAT〕

FATは各クラスタの使用状況を示すテーブルです. F-BASICはディスクの領域の管理を, このFATによりすべて行っています. FATはトラック0のセクタ27に書込まれ, 6～157バイトがクラスタの0～151に対応して, 各バイトは次の意味を持っています.

FATの値の示す意味

値(16進)	意　味(クラスタの使用状況)
0～97	このクラスタは使用中であり, 後続するクラスタを持つ. 又, その値が後続するクラスタ番号を示す.
C0～D9	このクラスタは使用中であり, ファイルの最後のクラスタである. 又, この値から16進のBFを引いた値は, クラスタ内の実際に使用されているセクタ数を示す.
FD	このクラスタ中の使用セクタはない.
FE	このクラスタはシステム領域である.
FF	このクラスタは未使用である.

〔ディレクトリ〕

ディレクトリは, トラック0のセクタ30～セクタ52に書込まれます. ディレクトリ部のセクタは32バイトずつの8ブロックに分割されます.

この32バイトは, ディレクトリスロットと呼ばれ, 次の構成になっています.

ディレクトリスロットの表わす内容

ディレクトリスロット内のバイト位置	内　　　容
0～7	ファイル名
8～10	リザーブ用
11	ファイルタイプ 00：BASICソース 01：BASICデータ 02：機械語ファイル
12	アスキーフラグ FF：アスキー形式で記録されている． 00：バイナリ
13	ランダム・アクセス・フラグ 00：シーケンシャル・ファイル FF：ランダム・ファイル
14	ファイル先頭クラスタ番号
15～31	リザーブ用

ディレクトリスロットは184個ありますので，最大184のファイルが登録可能ですが，クラスタ数は152なので，実際登録できるファイル数はユーザディスク使用時で152です．

〔ID〕

IDとは，システム識別記号領域とよばれ，下記に示すシステムに関する内容を記録しています．

トラック00のセクタ28

第0～2バイト ｛ "SYS"―システムディスクを示す．／"S␣␣"―ユーザディスクを示す．

第3バイト　：自動スタート機能で用いられる8インチのドライブ数．

第4バイト　：自動スタート機能で用いられるファイルバッファ数．
Haw many Disk Files（0～15）？で指定した数．

第5バイト　：自動スタート機能で用いられる5インチのドライブ数．

第6～255バイト　：未使用で全て"00"となっています．

第3章　F-BASICの命令

本章の見方

本章はF-BASICの命令をバージョンによらずすべて解説しております．各命令は〔機能〕,〔形式〕,〔バージョン〕,〔説明〕,〔例〕で構成されています．

各命令の右横にカッコで囲み，命令の読み方とフルネームを示しました．

〔例〕　　AUTO（オート：auto）

〔機　能〕

命令の概略を説明します．

〔形　式〕

命令の記述の仕方を示しています．記述には次のルールがあります．

（1）　各命令は英字で記述されていますが，大文字，小文字のいずれでも入力できます．但し，ダブルクォーテーション（〃）で囲まれた文字列は，大文字，小文字の区別をします．

（2）　角カッコ（[　]）は［　］内が省略可能を意味します．省略した場合は，BASICであらかじめ決められた値か，以前に指定した値が適用されます．

（3）　大カッコ（{　　}）は{　　}内のいずれかを選択することを示します．

（4）　省略記号（…）は，1行の許す範囲内で項目を繰返し指定できることを意味します．

〔例〕　READ　　変数名〔，変数名〕…の場合

READ　A，B，C，D

（5）　角カッコ以外の特殊記号で，左カッコ，右カッコ，コンマ（，），セミコロン（；），ハイフン（－），等価記号（＝），ダブルクォーテーション（〃）は指定された位置に正しく記入します．

（6）　␣は空白を示します．

（7）　3.4画面制御・グラフィック機能の各命令のなかで，パレットコードは，V1.0 V2.0ではカラーコードになります．

〔バージョン〕

各命令がF-BASICのどのバージョンで使用できるかを次のマークで示します．各マークが灰色であれば，該当のバージョンで使用できる命令であることを示しています．

V1.0 － F-BASIC V1.0で使用できます．

V2.0 － F-BASIC V2.0で使用できます．

V3.0 － F-BASIC V3.0で使用できます．

なお，命令の右横に（ディスク）と記してある命令は，V1.0 または V3.0 で，DISKモード時のみ有効な命令であることを示します．

〔説　明〕

各命令の詳細な解説と注意事項を記述しています．3.4画面制御・グラフィック機能の各命令のなかで，パレットコードは，V1.0，V2.0ではカラーコードになります．

〔例〕

サンプルプログラムを示します．

ROMモードとDISKモード

F-BASICには，使用できる命令によりROMモードとDISKモードがあります．但し，F-BASIC V2.0はROMモード，DISKモードの区別はなく，ミニフロッピィディスクまたは標準フロッピィディスクに使用できるコマンドがすべて格納されています．

ROMモードとは，本体に標準実装され，ROMにファームウェア化されたBASICに許される命令のみ使用できる状態をいいます．DISKモードとは，ファームウェア化されたBASICの命令と，ディスクからRAMに読込まれたBASICの命令が使用できる状態をいいます．

動作モード

本体に電源を入れるとBASICが起動され，タイトルを表示した後，〝Ready〞と表示されます．この〝Ready〞はBASICがコマンドレベルにあり，コマンドを受付け可能な状態にあることを示しています．このとき使用者は，直接モード，間接モードおよび，ターミナルモードのいずれかで使用することができます．

直接モードとは，BASICのコマンドあるいは，ステートメントを行番号をつけないで直接入力することで，このときBASICは，入力されたコマンドおよびステートメントをすぐに実行し，実行の終了後〝Ready〞と表示しコマンドレベルに戻ります．入力されたプログラムはメモリ上に残りません．

間接モードは，プログラムを入力するときに使用し，入力されたプログラムは，その時点では実行は行われず，BASICのプログラム領域に行番号順に格納されます．格納されたプログラムは，RUNコマンド，GOTO文あるいは，GOSUB文により実行することができます．なお，行の入力は，先頭に必ず行番号を付け，その後にコマンドあるいはステートメントを続けます．1つの行を入力後〝Ready〞の表示はされませんが，BASICはコマンドレベルにあります．

ターミナルモードは，他のコンピュータのターミナルとして使用する場合に用い，TERMコマンドを入力することによりターミナルモードになります．

3.1 コマンド

3.1.1 AUTO (オート：auto)

〔機　能〕

行の先頭に行番号を自動的に発生します．

〔形　式〕

AUTO　[行番号] [[,増分値]]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・行番号を最初の番号とし，増分値で増加させた行番号を付けます．
・行番号，増分値は，1～5桁の整数であり，指定のないときは10と見なされます．
・63999を越える行番号，あるいは，プログラム中にすでに存在する行番号を指定することはできません．
・CTRL＋C，CTRL＋X，BREAKキー（V1.0 V2.0ではSTOPキー）または，行番号の直後にリターンキーを入力することにより，AUTOの機能は終了して，BASICのコマンドレベルに戻ります．
・AUTOコマンドを用いてプログラムを入力している途中での修正はその行のみであって，すでに入力した行を修正することはできません．また，AUTOコマンドにより入力したプログラムに対して，直接スクリーンエディトすることはできません．AUTOの終了後，LISTコマンドにより修正しようとする行，または全ての行を一度画面に表示してから修正します．

〔例〕

```
AUTO 10
```

行番号は10を先頭に20，30……と10ステップで最後まで付けられます．

```
AUTO 200,45
```

行番号は200を先頭に245，290……と45ステップで最後まで付けられます．

```
AUTO ,30
```

行番号は10を先頭に40，70……と30ステップで最後まで付けられます．

AUTO

行番号は10を先頭に20, 30……と10ステップで最後まで付けられます.

AUTO 300,

直前に実行したAUTO文の増分値が20のときは，300， 320, 340, ……と付けられます.

3.1.2 DELETE （デリート：delete）

〔機　能〕

プログラムの行を削除します．

〔形　式〕

DELETE　[行番号1] [{, -}[行番号2]]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

- 行番号1と行番号2を指定したときは，その間に含まれる全ての行を削除します．
- 行番号1のみを指定したときは，その行だけを削除します．
- コンマまたは，ハイフォンと行番号2のみを指定したときは，行番号2までの全ての行を削除します．
- 行番号1とコンマまたは，ハイフォンを指定したときは，行番号1以上の行番号を持つ全ての行を削除します．
- コンマまたは，ハイフォンのみを指定したときは，全ての行を削除します．
- プログラムの中でDELETE文が現われたときは，実行後コマンドレベルに戻ります．

〔例〕

DELETE 40,500または**DELETE 40-500**

行番号40から500までの行を削除します．

DELETE ,200または**DELETE -200**

最初から行番号200までを削除します．

DELETE 120,または**DELETE 120-**

行番号120から最後までを削除します．

DELETE 95

行番号95を削除します．

DELETE

全ての行を削除します．

3.1.3 LIST （リスト：list）

〔形式1〕

```
LIST [行番号1] [{,/-}[行番号2]]
```

（省略形は　L.）

〔機　能〕

メモリ内にあるプログラムの全部または，一部を画面に出力します．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

- 行番号の指定のないとき，およびコンマまたはハイフォンのみの指定の場合は，プログラムの全てが表示されます．
- 行番号1のみが指定されたときは，その行だけが表示されます．
- 行番号1とコンマまたは，ハイフォンのみが指定されたときは，行番号1から最後の行までが表示されます．
- コンマまたは，ハイフォンと行番号2が指定されたときは，プログラムの最初の行から行番号2までが表示されます．
- 行番号1と行番号2が指定されたときは，その間の全ての行が表示されます．
- プログラムの実行中にエラーが起きたとき，行番号の代わりにピリオド（．）を指定すると，エラーの起きた文の行の内容が表示されます．
- プログラムの中でLIST文が用いられたときは，その文を実行後コマンドレベルに戻ります．

〔例〕

LIST

全ての行が表示されます.

LIST 50

行番号50の行のみ表示されます.

LIST 10-300またはLIST 10,300

行番号10から300の行が表示されます.

LIST -125またはLIST ,125

最初の行から125の行まで表示されます.

LIST 250-またはLIST 250,

行番号250の行から最後の行まで表示します.

〔形式 2〕

```
LIST "ファイルディスクリプタ"[,[行番号1]
     [{-,}[行番号2]]]
```

〔機　能〕

メモリ内にあるプログラムの全部または，一部を指定されたファイルに出力します．
行番号の指定は形式1と同じです．
デバイス名にカセットテープ，ディスク，あるいはバブルカセットを指定すると，アスキー形式のSAVEと同じになります．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイルディスクリプタを文字式，または文字変数で指定するときは，必ず引用符（"）で始めます．従って，文字変数で始まる文字式，または1つの文字変数で指定するときは，その前に空文字列を付加します．

〔例〕

```
LIST "LPT0:" 300,1250
```

プリンタに300行から1250行をプリンタに出力します．

```
LIST "CAS0:"
```

カセットテープに全ての行を出力します．

3.1.4 UNLIST (アンリスト：unlist)

〔機　能〕

非表示行番号を指定します.

〔形　式〕

```
UNLIST　行番号
```

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・このコマンド実行後は，指定された行番号より大きい行番号を持つ行の表示を行いません．UNLIST を一度実行すると，以後，指定した行以降をリストすることはできません．また，UNLIST を実行後プログラムを SAVE するときは，必ずバイナリ形式で行います．アスキー形式で SAVE すると，UNLIST の実行されている行に対しては SAVE できません．

〔例〕

```
10 UNLIST 20
20 A=10
30 B=20
40 C=A+B
50 PRINT C
60 END

Ready
RUN
 30

Ready
LIST

10 UNLIST 20

Ready
```

行番号 10 の UNLIST が実行されると20行以下のプログラムのリストが見れなくなります.

3.1.5 LLIST （エル・リスト：list to line printer）

〔機　能〕

メモリ内にあるプログラムの全部または一部をプリンタに印刷します．

〔形　式〕

LLIST　[行番号1] [{,_} [行番号2]]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・行番号の指定の方法は，LIST コマンドと全く同じです．単に LLIST を実行することは，LIST "LPTO：" を実行するのと全く同じです．

・LLIST 文がプログラムの中で用いられたときは，実行後コマンドレベルに戻ります．

〔例〕

LLIST

プリンタに全ての行を出力します．

LLIST 200

プリンタに行番号 200 の行のみ出力します．

LLIST 30-200 または **LLIST 30,200**

プリンタに行番号 30 から 200 行までを出力します．

LLIST -55 または **LLIST ,55**

プリンタに最初の行から 55 行まで出力します．

LLIST 360- または **LLIST 360,**

プリンタに行番号 360 の行から最後の行まで出力します．

3.1.6 RENUM （リナンバー：renumber）

〔機　能〕

プログラムの各行の行番号を付直します．

〔形　式〕

```
RENUM　[新行番号][,[旧行番号][,増分値]]
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・新行番号は，新しく付ける行番号の最初の番号です．省略値は，10です．
・旧行番号は，番号の付け替えが始められる現在のプログラムの中の行番号です．省略値は，プログラムの最初の行番号です．
・増分値は，新しく付ける行番号の増分量を示します．省略値は10です．
・RENUMコマンドは，GO TO，GOSUB，THEN，ON～GOTO，ON～GOSUB，ERLなどで参照している行番号も，新しい行番号に対応して変更します．
もし，これらの文で参照している行番号がプログラム中に存在しなかった場合は，"Undefined Line Number XXXXX in YYYYY"（XXXXXは存在しなかった行番号，YYYYYはその文の新しい行番号）のエラーメッセージが出力され，誤まっていた行番号はそのまま残ります．
・プログラムの中でRENUM文を実行したときは，その後コマンドレベルに戻ります．
・RENUMコマンドにより，63999を越える行番号を発生することはできません．また，プログラムの順序が変わるような指定をしてはなりません．（例えば，10，20，30の3つの行があるとき，RENUM　15，30を実行するような場合）．
このようなときには，"Illegal function call"のエラーになります．

〔例〕

```
RENUM
```

プログラム全体の行番号を付け直します．新しい行番号は10から10ステップで最後まで付けられます．

```
RENUM 30,50,20
```

行番号50に新行番号として30が付けられ，以降50，70……と20ステップで行番号が最後まで付けられます．

```
RENUM ,20,10
```

行番号20に新行番号として10が付けられ，以降20，30……と10ステップで行番号が最後まで付けられます．

RENUM 20,,20

最初の行から行番号が20，40，60……と20ステップで最後まで付けられます．

RENUM 30,15

行番号15に新行番号として30が付けられ，以降40，50……と10ステップで最後まで付けられます．

RENUM ,,30

プログラムの最初の行に行番号10が付けられ，以降40，70……と30ステップで最後まで付けられます．

RENUM ,50

行番号50に新行番号として10が付けられ，以降20，30……と10ステップで最後まで付けられます．

3.1.7 NEW （ニュー：new）

〔機 能〕

メモリにあるプログラムを消去し，全ての変数を初期化します．

〔形 式〕

```
NEW
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

NEW コマンドは，新しいプログラムを入力する前にメモリ内のプログラムをクリアするために使われます．NEW を実行後は，BASIC はいつもコマンドレベルに戻ります．NEW コマンドは，そのときオープンされているファイルがあるならば，そのファイルを全てクローズします．

〔例〕

```
10 FOR I=1 TO 9
20 FOR J=1 TO 9
30 PRINT I;"*";J;"=";J*I
40 NEXT J
50 PRINT
70 NEXT I
80 END


Ready
NEW

Ready
LIST


Ready
```

3.1.8 CLEAR（クリア：clear）

〔機　能〕

全ての数値変数を0に，全ての文字変数を空文字列に初期設定し，オペランドの指定により文字領域とBASICの使用する上限を設定します．

〔形　式〕

CLEAR　[文字領域の大きさ][,BASICの使用する上限のメモリアドレス]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・CLEARは現在メモリ内にあるプログラムを残した状態でデータとして使われているすべてのメモリを解放します．
CLEARを実行すると，数値変数は0に，文字変数は空文字列に初期設定され，配列変数は消去されます．
また，DEF文（DEFFN，DEFUSR，DEFINT，DEFDBL，DEFSNG，DEFSTR）により定義されたすべての情報は無効になります．

・BASICの起動時，文字領域の大きさは300バイトに初期設定されていますが，プログラムの実行中に〝Out of String Space〞のエラーが起きたときには，CLEARにより文字領域の大きさを300バイト以上に設定します．

・第2オペランドでは，BASICが使用できる上限のメモリアドレスを指定します．
BASICのプログラムから機械語サブルーチンを呼び出すときは，CLEARにより，BASICの使用する上限のメモリアドレスを指定し，機械語プログラムを格納する領域を確保する必要があります．

〔例〕

```
CLEAR 300,&H54FF
```

文字領域を300バイトに設定し，BASICの使用する上限アドレスを&H54FFに設定します．

```
CLEAR ,&H3000
```

文字領域はこれ以前に設定した値のままで，BASICの使用する上限アドレスのみ&H3000に設定します．

CLEAR 1000

文字領域を1000バイトに設定し，BASICの使用する上限アドレスは，これ以前に設定されている値になります．

CLEAR

文字領域，BASICの使用する上限アドレスは初期設定されます．

3.1.9 CONT （コント：continue）

〔機　能〕

BREAK キー（ (V1.0) (V2.0) では STOP キー）入力後または，STOP，END 文実行後，停止しているプログラムの実行を再開します．

〔形　式〕

```
CONT
```

（省略形は　C.）

〔バージョン〕　(V1.0)　(V2.0)　(V3.0)

〔説　明〕

・プログラムの実行停止後，直接モードで CONT コマンドを入力すると，停止した次の文から実行が再開されます．ただし，INPUT または，GCURSOR 文で実行が停止したときは，その文の初めから実行が再開されます．

・実行の停止後，直接モード文により中間結果を調べたり，変数の値を変更したりすることができます．その後 CONT または GOTO により，プログラムの実行を再開することが可能です．
ただし，プログラムの中断中，プログラムの変更を行なったときは，CONT により実行を再開することはできません，

・プリンタに出力中，又はカセットの入出力中に BREAK キー（ (V1.0) (V2.0) では STOP キー）が押されたときは，CONT により実行を再開することはできません．

〔例〕

```
10 A=123
20 B=456
30 C=A+B*10
40 STOP
50 PRINT C
60 END

Ready
RUN
Break In 40
Ready
CONT
 4683

Ready
```

3.1.10 RUN （ラン：run）

〔形式 1〕

```
RUN [行番号]
```

（省略形は R.）

〔機 能〕

メモリにあるプログラムの実行を始めます.

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・行番号が指定されたときは，その行から実行が始まります．また，行番号の指定がないときは，一番小さい行番号の行から実行が始まります.
このコマンドは，プログラムの実行前に全ての数値変数を 0 に，文字変数を空文字列に初期化し，オープンされている全てのファイルをクローズします.

・プログラムの実行は，BREAK キー（V1.0 V2.0 では STOP キー）の入力，あるいは END 文，STOP 文の実行により終了し BASIC のコマンドレベルに戻ります.

〔例〕

RUN

プログラムの最初から実行します.

RUN 50

行番号 50 から実行します.

〔形式2〕

```
RUN "ファイルディスクリプタ"[,R]
```

〔機　能〕

ディスクまたはカセット上のファイルからプログラムをロードし実行します.

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・RUNコマンドは, オープン状態にある全てのファイルのクローズおよび, メモリ内容の初期化を行なってから, プログラムをメモリにロードし, その後プログラムを実行します.
　ただし, Rの指定があるときは, オープンされているファイルのクローズは行いません.

・ファイルディスクリプタを文字式, または文字変数で指定するときは, 必ず引用符(")で始めます.

〔例〕

```
RUN "0:PROG"
```

フロッピィディスクのドライブ0から, PROGというファイル名を持つプログラムをメモリへロードし実行します.

```
RUN "CAS0:PROG1"
```

カセットテープから, PROG 1というファイル名を持つプログラムをメモリにロードし, 実行します.

3.1.11 LOAD (ロード：load)

〔機　能〕

プログラムをメモリにロードします．

〔形　式〕

LOAD "ファイルディスクリプタ"[,R]

（省略形は　LO.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイルディスクリプタで指定されるファイルからプログラムをメモリ上にロードします．このとき，オープンされているファイルがあるならば，その全てのファイルをクローズし，メモリの内容を初期化してからプログラムをメモリにロードします．
ただし，Rの指定があるときは，オープンされているファイルのクローズを行わないで，プログラムの実行を自動的に始めます．

〔例〕

```
LOAD "CAS0:EX1"
```

カセットテープから，EX1 というファイル名を持つプログラムをメモリへロードします．

```
LOAD "0:EX2"
```

フロッピィディスク0から，EX2 というファイル名を持つプログラムをメモリへロードします．

```
LOAD "1:EX3",R
```

フロッピィディスク1から，EX3 というファイル名を持つプログラムをメモリへロードし，実行します．

3.1.12 LOAD？ （ロードクエスション：load？）

〔機　能〕

カセットテープ上のチェックサムの照合を行います．

〔形　式〕

LOAD？　［"［CAS 0：］ファイル名"］

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・テープの上のプログラムのチェックサムと，SAVE 時に登録されたチェックサムとを比較照合します．一致が取れなかったときは，"Device I/O Error"が出力されます．
・ファイルディスクリプタの指定を省略したときは，テープ上の先頭のファイルに対して比較照合が行われます．
・ROM モードのときは，CAS 0：を省略できます．

〔例〕

```
LOAD? "EX1"
```

カセットテープ上で，EX 1 というファイル名を持つプログラムのチェックサムの照合を行います．

3.1.13 SAVE (セーブ：save)

〔機　能〕

メモリ内にあるプログラムをファイルに格納します．

〔形　式〕

SAVE　"ファイルディスクリプタ"[,{A / p}]

（省略形は　SA.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイルディスクリプタにより指定されるファイルにメモリ内のプログラムを格納します．
ファイルディスクリプタで指定されたデバイスが，フロッピィディスクの場合には，そのファイル名は媒体上に存在しない新たなファイルでなければなりませんが，そうでないときには次の処理が行われます．
直接モードで実行しているとき，
　Are you sure（Y　or　N）？
のメッセージが出力されます．このとき，Yと入力すると既に存在しているファイルは削除され，新たなファイルとして登録されます．Nと入力すると，既に存在しているファイルはそのまま保存され，メモリ内のプログラムのセーブは行われません．
間接モードで実行しているとき，
　File Already Exists In　行番号
のメッセージを出力して，コマンドレベルに戻ります．

・Aオプションの指定をすると，プログラムはアスキー形式（文字形式）でセーブされます．Aの指定がないときは，プログラムはバイナリ形式（内部コード化されたプログラム）でセーブされます．
バイナリ形式でセーブすると，セーブ時間が短くなり，ファイル容量も少なくてすみますが，セーブしたプログラムをMERGEコマンドで使用するときには，そのファイルはアスキー形式でセーブされていなければなりません．
また，アスキー形式でセーブされたファイルは，そのままデータファイルとして読むことができます．

・Pの指定をすると，プログラムの内容が保護されます．Pオプションを指定してセーブされたファイルは，メモリ上にロードしてLISTやEDITコマンドにより内容を見たり，行の削除を行おうとすると，"Protected Program"のエラーになります．
一度Pオプションの指定によりファイルが保護されると，これを解除する方法は準備されていませんので，指定するときには充分な注意が必要です．

〔例〕

```
SAVE "CAS0:PROG1"
```

メモリ内にあるプログラムをPROG1というファイル名で，カセットテープにバイナリ形式でセーブします．

```
SAVE "0:PROG2",A
```

メモリ内にあるプログラムをPROG2というファイル名で，フロッピィディスク0にマスキー形式でセーブします．

```
SAVE "1:PROG2",P
```

メモリ内にあるプログラムをPROG2というファイル名で，フロッピィディスク1にバイナリ形式で，プロテクトをかけてセーブします．

3.1.14 FILES (ファイルズ：files)

〔機 能〕

指定されたデバイスのディレクトリリストを出力します．

〔形 式〕

```
FILES [〝デバイス名〃] [,L]
```

〔バージョン〕 (V1.0) (V2.0) (V3.0)

〔説 明〕

・ディレクトリリストは，次の書式で出力されます．

ファイル名　種類　形式　編成　サイズ

種類は，0，1，2で表示され，次のことを意味します．

0 —— BASICプログラムファイル

1 —— データファイル

2 —— 機械語プログラムファイル

形式は，AまたはBで表示され，次のことを意味します．

A —— アスキー形式

B —— バイナリ形式

編成は，SまたはRで表示され，次のことを意味します．

S —— シーケンシャルファイル

R —— ランダムファイル

サイズはそのファイルの占める大きさを表し，バブルカセットの場合はページ数で，フロッピィディスクの場合はクラスタ数で示されます．カセットテープの場合はファイル名と種類と形式だけが出力されます．

・Lの指定を行うと，ディレクトリリストがプリンタにも出力されます．

・カセットテープに対するFILES文の終了は，BREAKキー（(V1.0) (V2.0) ではSTOPキー）を使います．

〔例〕

```
FILES "CAS0:"
```

カセットテープのディレクトリが次のように出力されます．

```
PROG1      0 A
PROG2      0 B
```

```
FILES"1:
```

フロッピィディスク1のディレクトリが次のように出力されます.

```
EX1        0 A S 1
EX2        0 B S 2
EX3        2 B S 1
EX4        1 A R 3

145 Clusters Free
```

3.1.15 NAME （ネーム：name） （ディスク）

〔機　能〕

フロッピィディスク上のファイル名を変更します．

〔形　式〕

NAME　〝旧ファイルディスクリプタ〟AS 〝新ファイルディスクリプタ〟

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・2つのファイルディスクリプタのデバイス名は，同じでなければなりません．

〔例〕

```
NAME "1:PROG1" AS "1:PROG2"
```

フロッピィディスク1のPROG1というファイル名をPROG2に変えます．

3.1.16 KILL (キル：kill)

〔機　能〕

フロッピィディスクまたは，バブルカセット上のファイルを削除します．

〔形　式〕

KILL "ファイルディスクリプタ"

〔バージョン〕 (V1.0) (V2.0) (V3.0)

〔説　明〕

・直接モードでこのコマンドを入力したとき，"Are you sure (Y or N)?"の問合わせのメッセージが出力されますので，削除するときはY，実行を取消すときはNを入力します．なお，間接モードでKILLを使用したときは，このメッセージは出力されず指定されたファイルが自動的に削除されます．

・KILLしようとするファイルは，クローズ状態になければなりません．オープン状態にあるファイルに対して，KILLを実行すると "File Already Open" のエラーになります．

・(V3.0)ではDISKモード時に有効です．

〔例〕

```
KILL "0:EX2"
```

フロッピィディスク0のEX2というファイルを削除します．

3.1.17 MERGE (マージ：merge)

〔機　能〕

メモリにあるプログラムと，指定されたファイルのプログラムを混合わせます．

〔形　式〕

MERGE 〝ファイルディスクリプタ〞[,R]

(省略形は　ME.)

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ファイルの中に，メモリ上のプログラムと同じ行番号を持つものがあれば，メモリ上の行が対応するファイル上の行と置換えられます．

・ファイルは，アスキー形式でセーブされていなければなりません．
もしバイナリ形式のファイルをMERGEすると〝Bad File Mode〞のエラーになります．

・Rの指定を行うと，MERGEの終了後プログラムの先頭から実行を開始します．オープンされているファイルのクローズは行なわれません．

〔例〕

```
MERGE "0:EX3"
```

メモリの中に次のプログラムが格納されているとします．

```
10 A=123
20 B=456
30 S=A+B
50 PRINT S
```

フロッピィディスク0のファイル名EX3の内容が次のとき，

```
20 B=789
40 S1=A-B
50 PRINT S,S1
60 END
```

MERGE実行後のメモリ内のプログラムは次のようになります．

```
10 A=123
20 B=789
30 S=A+B
40 S1=A-B
50 PRINT S,S1
60 END
```

3.1.18 SKIPF （スキップ・エフ：skip file）

〔機　能〕

指定したファイルの次にカセットテープを進めます．

〔形　式〕

SKIPF　[″[CAS 0：] ファイル名″]

（省略形は　SK.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイルのスキップを行うとき，ファイル上のチェックサムの照合を行います．
　チェックサムの値が正しくなければ，″Device I/O Error″になります．
・ファイルディスクリプタの指定を省略したときは，先頭のファイルの次にカセットテープを進めます．

〔例〕

SKIPF "DEMO"

ファイル名DEMOの次にカセットテープを進めます．

3.1.19 DSKINI （ディスク・アイ・エヌ・アイ：disk initialize） （ディスク）

〔機　能〕

フロッピィディスクのディレクトリの初期化を行います．

〔形　式〕

DSKINI　ドライブ番号

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ドライブ番号は0～3であり，初期化を行うディスクの入っているドライブを示します．

・このコマンドを直接モードで実行すると，〝Are you sure（Y or N）?″の問合わせのメッセージが出力されますので，初期化を行うときはYを入力し，コマンドの実行を取消す場合はNを入力します．

〔例〕

```
FILES "1:"

WRITE      0 B S 1
URIAGE     0 A S 6

145 Clusters free



DSKINI 1
Are you sure(Y or N)? Y

Ready
FILES "1:"

152 Clusters Free

Ready
```

3.1.20 BUBINI （バブ・アイ・エヌ・アイ：bubble initialize）

〔機　能〕

バブルカセットのディレクトリの初期化を行います．

〔形　式〕

BUBINI　ユニット番号

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ユニット番号は0または1で，初期化を行うバブルカセットの入っているユニットを示します．

・このコマンドを入力すると，〝Are　you　sure（Y　or　N）？〃のメッセージが出力されますので，初期化を行うときはYを入力し，実行を取消す場合はNを入力します．

〔例〕

```
FILES "BUB0:"

DATA          1 A S 11
EX1           0 B S 48

964 Pages Free



Ready
BUBINI 0
Are you sure(Y or N)? Y

Ready
FILES "BUB0:"

1023 Pages Free

Ready
```

3.1.21 EXEC（エグゼック：execute）

〔機　能〕

機械語プログラムを実行します．

〔形　式〕

```
EXEC [開始番地]
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・メモリ上にある機械語プログラムを，［開始番地］で指定されたアドレスから実行します．開始番地の指定がないときは，直前に実行されたLOADMコマンドの入口番地，または直前のEXECコマンドの開始番地から実行されます．

・EXECにより呼び出されるサブルーチンは，機械語のRTS命令を実行することにより，EXECの次の文に戻ります．

〔例〕

```
EXEC &H5000
```

&H5000番地からの機械語プログラムを実行します．

```
10 EXEC &H5000
20 ・・・
```

&H5000番地からの機械語プログラムがサブルーチンであれば，&H5000番地からの機械語プログラム実行後，行番号20へ戻ります．

3.1.22 LOADM (ロード・エム：load machine program)

〔機　能〕

機械語プログラムファイルをメモリにロードし実行します.

〔形　式〕

```
LOADM "ファイルディスクリプタ"
      [,[オフセット値][,R]]
```

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・ロードしようとするファイルは，SAVEMコマンドで作成した機械語ファイルでなければなりません.

・オフセット値を指定すると，そのファイルがSAVEMコマンドで作られたときに指定された開始番地に，オフセット値を加えた番地にロードされます。
したがって，このときその機械語プログラムは位置独立に作成されてなければなりません.
オフセット値を省略すると，そのファイルがセーブされたときに指定された開始番地と同じ番地にロードされます.

・LOADMにより機械語プログラムをロードするときは，必ずCLEAR文でBASICで使用する上限アドレスを設定し，機械語プログラムのメモリ領域を確保する必要があります.

・Rの指定をすると，プログラムをメモリにロードした後，入口番地から実行を始めます．Rの指定がないときは，プログラムをメモリへロードした後，BASICのコマンドレベルに戻ります.

〔例〕

```
LOADM "CAS0:SUB1",&H500
```

カセットテープからファイル名SUB 1と付けられたプログラムを，&H 500バイト加えたアドレスへロードします.

```
LOADM "0:SUB2",,R
```

フロッピィディスク0からファイル名SUB 2と付けられたプログラムを，メモリへロードし実行します.

3.1.23 SAVEM （セーブ・エム：save machine program）

〔機　能〕

メモリの内容をファイルにセーブします．

〔形　式〕

SAVEM　"ファイルディスクリプタ"，開始番地， 終了番地，入口番地

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・指定されたファイルに機械語プログラムを格納します．
・開始番地，終了番地によりセーブするメモリの範囲を指定します．
・入口番地により，このプログラムの入口点を指定します．この番地は，LOADMコマンドにより，このプログラムがメモリへロードされたとき，実行を開始する番地となります．
・ファイルデイスクリプタで指定されたデバイス名がフロッピイデイスクである場合には，そのファイル名は媒体上に存在しない新たなファイルでなければなりませんが，既に存在するファイルを指定したときには次の処理が行われます．
直接モードで実行しているとき，

Are you　sure (Y or N)?

のメッセージが出されます．このとき，Yと入力すると既に存在していたファイルは削除され，新たなファイルとして登録されます．Nと入力すると，既に存在していたファイルはそのまま保存され，メモリ内のプログラムのセーブは行われません．
間接モードで実行しているとき，

File Already Exists In 行番号

のメッセージが出力され，コマンドレベルに戻ります．

〔例〕

```
10 SAVEM"CAS0:OBJ",&H5000,&H5100,&H5000
```

&H 5000番地から&H5100番地の機械語プログラムを，OBJというファイル名でカセットテープにセーブします．このプログラムの実行開始アドレスは&H5000番地です．

3.1.24 HARDC（ハードコピー：hard copy）

〔機　能〕

画面データをプリンタにハードコピーします．

〔形　式〕

```
HARDC [{0|1|2}]
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・オペランドで0，1または2を指定することによりハードコピーの種類を選択することができます．

(1) 0またはオペランドのないとき

画面に表示されているキャラクタだけをプリンタにハードコピーします．

(2) 1のとき

画面上の1ドットをプリンタの横方向4ドットに拡大してプリントします．このとき，プリンタには黒，灰，白の濃淡レベルでプリントされます．画面上のドットの色とプリンタの濃淡レベルの対応は次のようになります．

画面上の青，赤，緑，白のドットは，黒でプリント
画面上の紫，水色，黄色のドットは，灰でプリント
画面上の黒のドットは，プリントされません．

(3) 2のとき

画面上の1ドットをプリンタの1ドットでプリントします．このとき，画面上の黒以外のドットは，黒でプリントされます．

〔例〕

```
HARDC 2
```

画面上の1ドットをプリンタの1ドットでプリントします．

3.1.25 MON （モニタ：monitor）

〔機　能〕

BASICのコマンドレベルからモニタコマンドレベルに移ります．

〔形　式〕

```
MON
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・モニタコマンドレベルからBASICコマンドレベルに戻るには，BREAKキー（V1.0 V2.0ではSTOPキー），CTRL－CまたはCTRL－Xを入力します．
・モニタコマンドレベルに入ると，プロンプトシンボル〝*〞が表示され，以下のサブコマンドの入力ができます．

コマンド名	機　能
M	メモリの内容を変更します．
G	指定アドレスに分岐します．
R	レジスタの内容を表示し，変更を可能にします．
D	指定アドレスから64バイトの内容を表示します．

（1） M

〔形　式〕

```
M[アドレス]
```

〔説　明〕

・Mに続けて1～4桁の16進数からなるアドレスを入力すると，そのアドレスの内容が表示され，入力待ちになります．内容を変更するときは，1～2桁の16進数を入力し，変更しないときは，リターンキーのみを入力します．以後，同じ操作を繰返します．

アドレスの指定を省略したときは，直前に用いたアドレスが適用されます．

アドレスの指定で，16進数以外を指定すると，モニタコマンドレベルに戻ります．

〔例〕

```
MON

*M5000
5000 00-20
5001 00-09
5002 00-17
5003 00-X

*
```

(2) G

〔形　式〕

G [分岐アドレス]

〔説　明〕

・分岐アドレスは，1～4桁の16進数で指定します．分岐アドレスを省略したときは，Rサブコマンドで設定したPCの内容により分岐します．

〔例〕

```
MON

*G5130
```

&H5130番地からの機械語プログラムを実行します．

(3) R

〔形　式〕

R

〔説　明〕

・MPUレジスタの内容を表示し，指定により内容を変更します．レジスタはCC, A, B, DP, X, Y, U, PCの順に表示されます．変更するときの方法は，Mサブコマンドと同じです．

〔例〕

```
MON

*R
CC B4-
A  00-
B  44-
DP 00-
X  AFA2-5649
Y  C830-6517
U  0340-
PC AFA7-

*
```

（4） D

〔形　式〕

D［アドレス］

〔説　明〕

・指定アドレスから64バイトの内容を表示します．アドレスの指定を省略したときは，直前に表示されたアドレスの次のアドレスから，64バイトの内容が表示されます．

〔例〕

```
MON

*D9000

9000 E1 96 E1 9C D9 A1 CD BE
9008 D0 14 D1 F5 D1 ED CA A5
9010 D1 6C D1 37 CE 8B D6 BD
9018 DD 90 E6 24 DD 2E CB 03
9020 E2 56 AB 74 9E BD 9F 0D
9028 94 AE CF D7 CA B1 BF 4D
9030 B0 F5 B1 23 93 F8 E9 53
9038 E9 6E E6 B6 DE E3 DF 91
*
```

3.1.26 TERM (ターミナル：terminal)

〔機 能〕

動作モードをターミナルモードにします．

〔形 式〕

```
TERM ["cbpsma"]
```

〔バージョン〕 (V1.0) (V2.0) (V3.0)

〔説 明〕

このコマンドを実行すると，RS－232C 通信ポートを介して外部機器のターミナルとして使用することができます．オペランドにより次の項目を指定できます．

・c クロックを指定します．
  S：Slow クロック (1/64 分周)，F：Fast クロック (1/16 分周)
  Slow クロックのとき，ボーレートに 300，600，1200，2400，4800 の選択をすることができます． Fast クロックのときは， Slow クロックの 4 倍のボーレート ((V1.0) (V2.0) では 9600 まで) になります．なお，ボーレットの設定はディップスイッチにより行います．
・b データのビット長を指定します．
  8：8 ビット長，7：7 ビット長
  7 ビット長を指定したときは，SI (シフトイン)，SO (シフトアウト) コードが自動的に付けられ，英数字とカナの区別がされます．
・p パリティの形式，有無を指定します．
  N：パリティなし，O：奇数パリティ，E：偶数パリティ
・s ストップビットのビット数を指定します．
  2：2 ストップビット，1：1 ストップビット
・m 通信モードを指定します．
  F：全二重通信　H：半二重通信
・a オート LF を行うか否かを指定します．
  A：オート LF を行う，N：オート LF をしない．
  オート LF の選択を行うと，CR コードを受信したとき，自動的に LF コードが出力され，改行が行われます．

TERM コマンドのオペランドを省略したときは，TERM "S8N2FN" を入力したときと同じになります．

ターミナルモードから BASIC のコマンドモードに移るときは，BREAK キー((V1.0) (V2.0)では STOP キー) を入力します．

ターミナルモードのとき，ファンクションキーの6～10は次のように特別な意味を持っています．

PF 6：プリンタへのエコーバックをする／しない．

PF 7：全二重／半二重．

PF 8：コントロールコード（&H00～&H1F）を文字として表示する／しない．

PF 9：画面の文字をプリンタにハードコピーする．

PF 10：約100 ms間のBreak信号を送出します．

PF 6～PF 8は，一度押すと以前と逆の状態になります．なお，ターミナルモードになった直後には，次のように設定されています．

PF 6：プリンタへのエコーバックをしない．

PF 7：全二重

PF 8：コントロールコードの表示をしない．

〔例〕

TERM "S8N2FA"

RS232-Cポートは次のように設定されます．

・Slowクロック指定となり，ボーレートはディップスイッチにより，300，600，1200，2400，4800のいずれかを選択できます．

・データの形式は，データ長8ビット，ストップビット2ビット，パリティビットなしに設定されます．

CRコードを受信すると自動的に改行が行われます．

3.1.27 EDIT （エディット：edit）

〔機　能〕

指定された行を画面に表示します．

〔形　式〕

```
EDIT　行番号
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・EDIT 文を実行すると，画面がクリアされた後，指定された行が表示され，カーソルが行番号の直後で点滅します．以後次のカーソル移動キー，EDIT キーを使用して 1 行の編集を行うことができます．

[→]　カーソルを右へ移動します．

[←]　カーソルを左へ移動します．

[↑]　カーソルを上へ移動します．

[↓]　カーソルを下へ移動します．

[INS]　カーソルの示す位置に文字を挿入します．[INS] キーを押すと LED が点灯し，挿入可能となります．文字を 1 個挿入すると，カーソル以降のデータは全体に右へ 1 桁移動します．

[DEL]　カーソルの示す個所をまっ消し，カーソル以降のデータを左へつめます．

[EL]　カーソル以降の文字をまっ消します．

[⇦]　((V1.0)(V2.0)では [BS])カーソルの示す左隣りの文字をまっ消し，カーソル以降の文字を左へつめます．

リターンキーの押下により編集動作を終了し，BASIC のコマンドレベルへ戻ります．

〔例〕

```
EDIT 300
```

3.2 一般ステートメント

3.2.1 DEF FN (ディファイン・ファンクション：define function)

〔機　能〕

使用者の関数を定義し，それに名前を付けます．

〔形　式〕

DEF FN　名前[(パラメータリスト)]＝式

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

- 名前は数値変数名または文字変数名で，FN＋名前が関数名になります．
- パラメータリストは，その関数が呼ばれたときに実際の値と置換えられる変数名の並びで，変数名の間はコンマで区切ります．
  パラメータリストの変数は，関数が呼び出されるときに与えられる式（実引数）と1対1に対応します．
- 関数名の型と式の結果の型は一致しなければなりません（数値，文字）．
  また，パラメータリストの変数名の型は，対応する実引数の型と一致しなければなりません．
- DEF FN 文は，それが定義する関数が呼ばれる前に書かれていなければなりません．
  また，直接モードでDEF FN文を使うことはできません．
- パラメータリストに書かれた変数名と同じ変数名がプログラムの中の他の行で使われていてもかまいません．それらは，異なった変数として取扱われます．

〔例〕

```
10 DEF FNA(X,Y,Z)=X^3+Y^2-Z
20 A=FNA(5,2,15)
```

関数FNAのパラメータX，Y，Zにそれぞれ5，2，15が与えられます．

3.2.2 DEF USR（ディファイン・ユーザ：define user）

〔機　能〕

機械語プログラムの開始番地を指定します．

〔形　式〕

DEF　USR[数字]＝整数

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・数字は0から9の任意の数で，USR＋数字が関数名になり，その名前により関数が呼び出されます．なお，数字を省略すると0とみなされます．

・整数は，メモリ上に格納されている機械語プログラムの実行開始アドレスを指定します．機械語プログラムは，CLEARコマンドの第2オペランドで指定した番地から，F-BASICシステムコードの最下位番地の間に格納しなければなりません．

・USR関数による機械語プログラムの呼び出し方は，他の関数と同じであり，次の形式で行います．

USR[数字]（引数）

引数は文字式または数値式です．この引数の情報は，A－レジスタとX－レジスタにより機械語サブルーチンに渡されます．

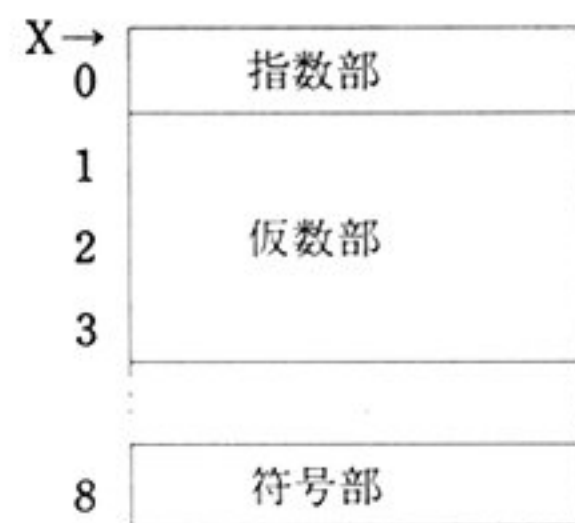
A－レジスタ＝2	引数の型が	整数型
3		文字型
4		単精度実数型
8		倍精度実数型

X－レジスタ　　引数の値が格納されている番地を示し，格納形式は以下のとおりです．（数値変数の場合，引数はワークエリア内の固定番地に転送されます．）

① 整数型のとき

X－レジスタ＋2，3番地に，2の補数表現された値が入ります．

② 単精度実数型のとき

X→	
0	指数部
1	仮数部
2	
3	
:	
8	符号部

指数部には，2の指数の値が入ります．&H80のげたはかせにより，指数の正負が表されます．すなわち&H 00～&H 80に対しては，2^{-128}～2^{0}を示し，&H 80～&H FFに対しては，2^{0}～2^{127}を示します．仮数部は符号なしの2進数で表現されます．

③ 倍精度実数型のとき

X→	
0	指数部
1	仮数部
2	
3	
4	
5	
6	
7	
8	符号部

符号部は，このデータの符号を表し，その最上位ビットが0のときは正，1のときは負を示します．

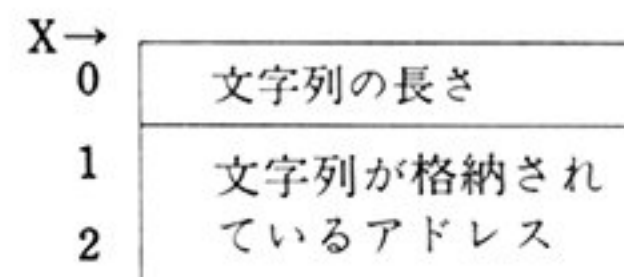
指数部をe，仮数部をm，符号部をSignで表せば，この形式で表現されるデータは次の式で表すことができます．

$$X = \mathrm{Sign}\ 0.m \times 2^{e}$$

④ 文字型のとき

このとき，X－レジスタはストリングディスクプリタと呼ばれる3バイトのデータの先頭番地を示します．

X→	
0	文字列の長さ
1	文字列が格納されているアドレス
2	

〔例〕

```
10 A%=56
20 DEF USR0=&H6100
30 B%=USR0(A%)
```

USR0を実行するとA－レジスタに2，X－レジスタに56が格納されているメモリアドレスがそれぞれセットされ，その後&H6100番地の機械語プログラムを実行します．

3.2.3 DEFINT/SNG/DBL/STR （ディファイン・イント/シングル/ダブル/ストリング：define integer／single／double／string）

〔機　能〕

変数の型を整数，単精度，倍精度または文字に宣言します．

〔形　式〕

DEF 型指定　文字の範囲

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・このDEF文は，文字の範囲で指定される文字で始まる全ての変数名の型を，型指定で示される型に宣言します．

・型指定により，変数の型は次のように宣言されます．

DEFINT　変数を整数型に宣言します．
DEFSNG　変数を単精度型に宣言します．
DEFDBL　変数を倍精度型に宣言します．
DEFSTR　変数を文字型に宣言します．

・文字の範囲は，1字の英字，または1字の英字－1字の英字により指定します．後者の場合，最初の英字は，後の英字よりもアルファベット順で上位のものでなければなりません．

・型宣言文字をもつ変数は，その型宣言文字がDEF型指定文に優先します．

・型宣言文がなく，型宣言文字も伴わない変数は，全て単精度変数として扱われます．

〔例〕

```
DEFINT A,C,E
```

A，C，Eで始まる全ての変数を整数変数に宣言します．

```
DEFSNG B-X
```

BからXまでの文字で始まる全ての変数を単精度変数に宣言します．

```
DEFDBL D-G
```

DからGまでの文字で始まる全ての変数を倍精度変数に宣言します．

```
DEFSTR A,E-G
```

A，E，F，Gで始まる全ての変数を文字変数に宣言します．

3.2.4 REM (リマーク：remark)

〔機　能〕

プログラムの中の注釈です．

〔形　式〕

REM　注釈文字列

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・REM 文はプログラムの中の注釈であって，実行には影響をおよぼしません．
・キーワード「REM」の代わりにシングルクォート（'）により，REMの代用をすることができます．
・REM 文では，コロン（：）も注釈の一部と見なされます．従って，コロン（：）で区切って次の文を続けることはできません．

〔例〕

```
10 REM PROGRAM No. 10
20 ' DATE 1981,10,4
```

3.2.5 END (エンド：end)

〔機　能〕

プログラムの実行を終了し，全てのオープンされているファイルをクローズした後，コマンドレベルに戻ります．

〔形　式〕

```
END
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・END文はプログラムの実行を終了させるために，プログラムの中のどこに置いてもかまいません．

・プログラムの最後には，END文はなくてもかまいませんが，この場合にはファイルのクローズは行われません．

〔例〕

```
10 A=123
20 B=456
30 C=A+B
40 PRINT C
50 END
```

3.2.6 FOR～NEXT（フォー・ネクスト：for～next）

〔機　能〕

一連の命令を繰返し実行します．

〔形　式〕

FOR　変数＝式１　TO　式２［STEP　式３］

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・変数は制御変数と呼ばれ，ループのカウンタとして使用されます．変数は倍精度型を除く数値変数でなければなりません．

・式１，式２および式３は数値式で，それぞれ初期値，終値および増分値を指定します．なお，増分値が省略されたときは，１と見なされます．

・実行は次のように行われます．
まず，式１の値を変数の初期値とし，FOR文から，それに対応するNEXTまでの文を実行します．NEXT文に達すると，変数に式３の値を加え，それを変数の新しい値とし，式２と比較します．もし変数の値の方が大きいときは，NEXT文に続く文を実行し，そうでなければ，FOR文の次の文に戻り，同じ処理を繰返します．これをFOR～NEXTループと呼びます．
式３の値が正のときは，以上のように判定が行われますが，式３の値が負の場合は，式１の値は式２よりも大きくなければなりません．このとき，カウンタの値が式２より小さくなるまでFOR～NEXTループが繰返されます．

・増分値が正で，初期値が終値より大きいとき，または増分値が負で，初期値が終値より小さいときは，FOR～NEXTループは一回だけ実行されます．

・ループの実行中に式１，式２，式３の値を変えてはいけません．

・多重ループについて
FOR～NEXTループは，入れ子構造にすることができます．すなわち，あるFOR～NEXTループの中に他のFOR～NEXTループを含めることが可能です．この場合，内側のFOR～NEXTループは，外側のループの中に完全に含まれていなければなりません．
多重ループを構成するときは，それぞれのループの制御変数は，他のループの制御変数名と異なっていなければなりません．

〔例〕

```
10 PI=3.14159
20 FOR I=0 TO 360 STEP 10
30 S=SIN(PI/180*I)
40 PRINT S
50 NEXT
60 END
```

SIN の値を 0 度から 360 度まで 10 度ごとに求めています.

```
10 M=&H20.
20 FOR I=&HFF TO M STEP -1
30 PRINT CHR$(I);
40 NEXT
50 END
```

増分値を負の数にして終値Mより小さくなるまでループを繰返します.

```
10 FOR I=1 TO 9
20 FOR J=1 TO 9
30 PRINT I;"*";J;"=";J*I
40 NEXT J
50 PRINT
70 NEXT I
80 END
```

1 * 1 から 9 * 9 までを表示しています. FOR～NEXT ループが二重になっていますので, 30 の行を 9 回繰返したところで, I が 1 つ増加します.

3.2.7 NEXT（ネクスト：next）

〔機　能〕

FOR～NEXT ループの終りを指定します．

〔形　式〕

NEXT　[変数名[,変数名]…]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・NEXT 文は，いまだ対応づけられていない最も直前の FOR 文と対応します．
・変数名は，対応する FOR 文の制御変数名と一致しなければなりません．
・1つの NEXT 文で複数個の FOR～NEXT ループを終了することができます．この場合，NEXT 文に終了しようとする FOR 文の制御変数名を順番に書かなければなりません．

〔例〕

3.2.6　FOR～NEXT 文の項を参照して下さい．

3.2.8 GOTO（ゴーツー：goto）

〔機　能〕

無条件に指定された行番号に分岐します．

〔形　式〕

GOTO　行番号

（省略形は　GO.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・指定された行番号を持つ文が実行文のときは，その文から実行が継続されます．
　非実行文のときは，その直後に続く文の最初の実行文から実行が継続されます．

〔例〕

```
10 INPUT A,B
20 PRINT A+B
30 PRINT A-B
40 PRINT A*B
50 PRINT A/B
60 GOTO 10
70 END
```

入力データの加減乗除を計算し，プリントします．60行のGO TO文により無条件に10行に分岐しますので，何回でも繰返して演算することができます．
このプログラムはBREAKキー（V1.0 V2.0ではSTOPキー）CTRL+C，CTRL+Xの入力によりBASICのコマンドレベルに戻ります．

3.2.9 ON～GOTO (オン・ゴーツー：on～goto)

〔機　能〕

式の値により指定された行番号の1つに分岐します．

〔形　式〕

ON 式 GOTO 行番号[,行番号]…

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・式の値が1ならば1番目の行番号に，2ならば2番目の行番号に，というように式の値により分岐します．もし式の値が整数でないときは，小数部を四捨五入した整数に変換されます．

・式の値が0，または行番号の個数よりも大きいときは，次の文にプログラムの制御が移ります．

・左の値が負の場合は，Illegal Function Callエラーとなります．

〔例〕

```
10 INPUT A
20 ON A GOTO 30,50
30 B=1
40 GOTO 60
50 B=2
60 FOR I=1 TO B
70 BEEP
80 PRINT "ピー"
90 FOR J=1 TO 130
100 NEXT J
110 NEXT I
```

入力データが1であれば行番号30から実行し，2なら行番号50から実行します．

入力データが1の場合ブザーが1回，2の場合は2回鳴ります．

3. 2. 10 GOSUB （ゴー・サブ：go to subroutine）

〔機　能〕

サブルーチンを呼出し，サブルーチン終了後はGOSUB文の直後の文に戻ります．

〔形　式〕

GOSUB　行番号

（省略形は　GOS.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・行番号は，呼出すサブルーチンの入口の行番号でなければなりません．

〔例〕

```
10 N$="F-BASIC"
20 GOSUB 100
30 PRINT N$
40 END
100 PRINT N$
110 N$="Ver 3.0"
120 RETURN
```

GOSUB文により行番号100へ分岐し，RETURN文によりGOSUB文の次の行30へ戻ります．

3.2.11 RETURN (リターン：return)

〔機　能〕

サブルーチンを終了し，呼出したプログラムへ復帰します．

〔形　式〕

```
RETURN [行番号]
```

(省略形は　RET.)

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・RETURN 文は，サブルーチンの中のどこにあってもかまいません．また，複数個の RETURN 文を使用することもできます．

・行番号の指定のあるときは，その行に復帰します．行番号の指定がないときは，一番最近に実行された GOSUB 文の直後の文に復帰します．
割込み処理ルーチンでの RETURN 文は，ON ～文の項を参照して下さい．

〔例〕

3.2.10 GOSUB 文を参照して下さい．

3.2.12 ON～GOSUB（オン・ゴーサブ： on～go to subroutine）

〔機　能〕

式の値により指定された行番号をもつサブルーチンを呼出します．

〔形　式〕

ON　式　GOSUB　行番号[,行番号]…

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式の値が1ならば1番目の行番号に，2ならば2番目の行番号に，というように式の値により分岐します．もし式の値が整数でないときは，小数部を四捨五入した整数に変換されます

・式の値が，0または行番号の個数よりも大きいときは，次の文にプログラムの制御が移ります．左の値が負のときは，Illegal Function Callエラーとなります．

・指定された行番号は，サブルーチンの先頭の行番号でなければなりません．

〔例〕

```
10 S1=0
20 S2=0
30 INPUT A
40 ON A GOSUB 100,200,300
50 GOTO 30
60 END
100 'SUB 1
110 S1=S1+1
120 RETURN
200 S2=S2+1
210 RETURN
300 PRINT S1,S2
310 RETURN 60
```

入力したデータが1であれば行番号100のサブルーチン，2なら行番号200のサブルーチンを実行します．3のときは，行番号300のサブルーチンを実行し，RETURN文により行番号60に戻り実行を終了します．

3.2.13 STOP （ストップ：stop）

〔機　能〕

プログラムの実行を停止して，コマンドレベルに戻ります．

〔形　式〕

```
STOP
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・STOP文は，プログラムの実行を停止するために使用し，プログラムの中のどこに置いてもかまいません．

STOP文が実行されると，次のメッセージが出力されます．

Break In nnnnn

nnnnnはSTOP文の行番号を示します．

・END文と異なり，STOP文はファイルのクローズを行いません．

・STOP文が実行されると，BASICはコマンドレベルに戻りますが，CONTコマンドによりプログラムの実行を再開することができます．

〔例〕

```
10 S=0:S1=0
20 FOR I=1 TO 100
30 S=S+I:S1=S1+I^2
40 NEXT
50 PRINT S
60 STOP
70 PRINT S1
80 END
```

行番号60のSTOP文によりBASICのコマンドレベルに戻ります．次にCONTコマンドを入力すると，行番号70から実行が再開されます．

3.2.14 IF～THEN～ELSE（イフ・ゼン・エルス：if～then～else）

〔機　能〕

式の結果により実行すべき文を選択します．

〔形　式〕

```
IF 式 THEN {文|行番号}
      [ELSE {文|行番号}]
```

または

```
IF 式 GOTO 行番号[ELSE{文|行番号}]
```

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・式は論理式または関係式でなければなりません．

・式の値が真（値が0でない）のときは，THENまたはGOTO文が実行されます．THENの後には，実行する文または分岐する行番号を続けることができます．またGOTOの後には，分岐する文の行番号を続けます．

・式の値が偽（値が0）のときは，THENまたはGOTO文は無視されます．ELSEがある場合は，それが実行され，ELSEがなければ次の文に実行の制御が移ります．

・IF文のネスティング

IF文は，THENまたはELSEの後に，さらにIFを書くことにより入れ子構造にすることができます．もしIF文中のTHENとELSEの個数が異なるときは，それぞれのELSEは，最も近くにありまだ対応づけられていないTHENに対応します．

IF文のネスティングは，1行の範囲内で何重にでもすることができます．

〔例〕

```
10 INPUT A,B
20 IF A>B THEN A=B
30 PRINT A
40 END

Ready
RUN
? 3,7
 3

Ready
```

入力データA，Bのうち小さい方をプリントします．

```
10 INPUT A,B,C
20 IF A>B THEN IF A>C THEN PRINT A
   ELSE PRINT C ELSE IF B>C THEN
   PRINT B ELSE PRINT C
30 GOTO 10

Ready
RUN
? 9,6,20
 20
? 2,5,8
 8
?
```

3つの入力データの中で一番大きいものをプリントします.

3.2.15 WHILE～WEND (ホアイル・ダブルエンド：while～wend)

〔機　能〕

一連の命令を条件付きで繰返し実行します.

〔形　式〕

```
WHILE 式
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・式は論理式または関係式でなければなりません.

・WHILE 文とそれに対応する WENDの間の部分を WHILE～WENDループと呼びます.

・式の値が真（値が0でない）の間, WHILE～WEND ループが繰返し実行されます.
式の値が真でなくなったならば, WEND の次の文から実行が始まります.
式の値が最初から真でない場合は, ループは1回も実行されません.

・WHILE～WEND ループは, メモリの許される範囲で何重にでもネストできます.
それぞれの WEND 文は, 最も直前に実行された WHILE 文に対応づけられますが, WHILE と WEND の対応がつかなければ, "While Without Wend" 又は "Wend Without While" のエラーが発生します.

〔例〕

```
10 WHILE I<90
20 I=RND(1)*100
30 PRINT I
40 WEND
50 END

Ready
RUN
 59.1065
 20.7991
 55.0967
 63.5863
 10.411
 80.6334
 4.29073
 95.3862

Ready
```

Iの値が90より小さい間, WHILE～WEND ループを繰返します.

3.2.16 WEND（ダブル・エンド：wend）

〔機　能〕

WEND は，WHILE～WEND ループの終りを指定します．

〔形　式〕

```
WEND
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・WEND 文は，まだ対応づけられていない最も直前の WHILE と対応します．

〔例〕

3.2.15 WHILE 文を参照して下さい．

3.2.17 LET （レット：let）

〔機　能〕

右辺の式の結果を左辺の変数に代入します.

〔形　式〕

[LET]　変数名＝式

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・キーワード LET は省略することができます.

・右辺が数値式のときは，左辺は数値変数でなければなりません．また右辺が文字式の場合は，左辺は文字変数でなければなりません．一致していない場合には〝Type Mismatch〟のエラーになります.

〔例〕

```
10 LET A=5
20 LET B$="FUJITSU"
```

変数Aに5，B$に文字FUJITSUを代入します．LETを省略して次のように書くこともできます.

```
10 A=5
20 B$="FUJITSU"
```

3.2.18 SWAP（スワップ：swap）

〔機　能〕

2つの変数の値を交換します．

〔形　式〕

```
SWAP　変数，変数
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・変数の型は，整数，単精度，倍精度，文字のいずれであっても交換することができますが，2つの変数の型は同じでなければなりません．一致していない場合には，"Type Mismatch" のエラーになります．

・第2オペランドの変数が単純変数の場合には，その変数には値が定義されていなければなりません．

〔例〕

```
10 INPUT A,B
20 PRINT "A=";A,"B=";B
30 SWAP A,B
40 PRINT "A=";A,"B=";B
50 END

Ready
RUN
? 2,5
A= 2          B= 5
A= 5          B= 2

Ready
```

行番号30のSWAP実行により，変数A，Bの値は入れ換わります．

3.2.19 DIM (ディメンジョン：dimension)

〔機　能〕

配列変数の次元数と添字の最大値を指定し，その変数にメモリ領域を割当てます．

〔形　式〕

DIM　変数名（添字の最大値[,添字の最大値]…）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・カッコの中に書かれた添字の最大値の個数が，配列の次元数を示します．

・DIM 文は，指定された配列のすべての要素を 0 に初期設定します．

・DIM で配列の宣言をしないで，配列変数名が使われたときは，添字の最大値が 10 の配列が暗黙に宣言されます．

・配列変数を参照するとき，各次元の添字の値は DIM 文で宣言された添字の最大値よりも小さくなければなりません．また次元の数は DIM 文と合ってなければなりません．そうでなければ，"Subscript Out Of Range" のエラーになります．

〔例〕

```
10 DIM A(1,20)
```

2 次元配列を宣言します．

```
20 DIM L%(10)
```

整数形式の配列を宣言します．

```
30 DIM B$(40)
```

文字配列を宣言します．

3.2.20 POKE （ポーク：poke）

〔機 能〕

メモリの指定番地にデータを書込みます．

〔形 式〕

POKE 書込み番地，データ

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・書込み番地は，0～65535の範囲になければなりません．
・データは，0～255の範囲になければなりません．

〔例〕

```
10 POKE &H5000,57
```

＄5000番地に57を書込みます．

3.2.21 DATA （データ：data）

〔機　能〕

READ 文によって読込まれる数値および文字定数を格納します.

〔形　式〕

DATA　定数 [,定数]…

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

- DATA 文は非実行文であって，プログラムのどこにでも書くことができます.
- 1つの DATA 文には，コンマで区切って1行に入るだけの定数を書くことができ，またプログラムの中にいくつもの DATA 文を書くことができます．これらの DATA 文で定義された定数は，行番号の若い DATA 文から一連の連続したデータとしてとらえられます.
- 定数は，数値定数（整数，固定小数点，浮動小数点，16 進数，8 進数），文字定数のいずれであってもかまいません．ただし，文字定数の中にコンマ，コロン，または前後に意味のある空白を含むときは，その文字定数全体を引用符（"）で囲み，それ以外の場合は，引用符で囲む必要はありません.
- DATA 文で定義された定数は，READ 文により順々に読込まれますが，対応する変数と定数の型は一致していなければなりません.
- DATA 文で定義された定数は，RESTORE 文により再び最初から読むことができます.

〔例〕

```
10 READ A,B
20 IF A=0 THEN END
30 C=A+B
40 PRINT "A+B=";C
50 GOTO 10
60 DATA 10,20,23,5
70 DATA 30,24,0,0

Ready
RUN
A+B= 30
A+B= 28
A+B= 54

Ready
```

3.2.22 READ（リード：read）

〔機　能〕

DATA 文により定義された定数を変数に読込みます．

〔形　式〕

```
READ　変数名[,変数名]…
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・READ 文は DATA 文と共に使用し，DATA 文で定義された定数と，READ 文の変数を1対1に対応させながら読込みます．

・READ 文の変数は数値変数でも文字変数でもかまいませんが，対応する定数の型と一致しなければなりません．

・1つのプログラムの中で，指定された変数の数が DATA 文の定数の個数よりも多い場合は，"Out of Data" のエラーが出力されます．

・RESTORE 文を実行することにより，最初の DATA 文の定数から読むことができます．

〔例〕

3.2.21　DATA 文を参照してください．

3.2.23 RESTORE （リストア：restore）

〔機　能〕

DATA 文を最初から読むように指示します．

〔形　式〕

```
RESTORE [行番号]
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・RESTORE 文を実行すると，その次の READ 文はプログラムの中の最初の DATA 文から読み始めます．行番号を指定すると，その行以降の DATA 文から読み始めます．

〔例〕

```
10 READ A,B,C
20 READ D,E,F
30 PRINT "A=";A;"B=";B;"C=";C
40 PRINT "D=";D;"E=";E;"F=";F
50 PRINT
60 RESTORE 130
70 READ A,B,C
80 RESTORE
90 READ D,E,F
100 PRINT "A=";A;"B=";B;"C=";C
110 PRINT "D=";D;"E=";E;"F=";F
120 DATA 1,3,5
130 DATA 2,4,6

Ready
RUN
A= 1 B= 3 C= 5
D= 2 E= 4 F= 6

A= 2 B= 4 C= 6
D= 1 E= 3 F= 5

Ready
```

行番号60の RESTORE 文により 70行の READ 文は 130行の DATA 文から読み始めます．また，80行の RESTORE文により，90行の READ 文は 120行の DATA 文から読み始めます．

3.2.24 LSET, RSET (エル・セット：left set／アール・セット：right set)　(ディスク)

〔機　能〕

文字データをランダムファイルバッファに移します.

〔形　式〕

```
{LSET}
{RSET}  文字変数＝式
```

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

- LSET, RSET の実行に先立って, FIELD 文が実行されていなければなりません.
- 文字変数は, FIELD 文で割当てられた変数名です.
- 式は文字式であり, その値が文字変数に代入されます.
- 式の結果, 文字列がFIELD 文で指定された長さよりも短い場合, LSET 文は, そのフィールドに左づめで, RSET 文は右づめでデータを満し, 残った部分には空白がつめられます.
  もし, 式の方が長い場合には, 右側から文字が捨てられます.
- 数値を LSET または RSET により文字変数により代入するためには, 関数 MKI$, MKS$, MKD$ により文字形式に変換しなければなりません.

〔例〕

```
10 OPEN "R",#1,"TABLE"
20 FIELD #1,20 AS A$,20 AS B$
30 INPUT "DATA";C$,D$
40 LSET A$=C$
50 RSET B$=D$
60 PUT #1,1
70 CLOSE #1
80 END
```

3.2.25 RANDOMIZE （ランダマイズ：randomize）

〔機　能〕

乱数の系列を変更します．

〔形　式〕

```
RANDOMIZE [式]
```

（省略形は　RNDM.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式であり，その結果の値は－32768～32767 でなければなりません．
・式が省略されたときは，プログラムの実行が停止し，次の表示が行われます．

Random Number Seed (－32768 to 32767)?

この時，－32768～32767 の範囲の値を入力すると，実行が再開されます．
・乱数の系列を変えなければ，関数 RND はプログラムが RUN するたびに同じ系列の乱数を繰返します．

〔例〕

```
10 RANDOMIZE
20 FOR I=1 TO 10
30 A=INT(RND(1)*8)+1
40 PRINT A;
50 NEXT
60 END

Ready
RUN
Random Number Seed (-32768 to 32767)? 5649
 2  3  5  1  1  8  3  5  2  3
Ready
```

1～9の乱数を発生しています．行番号 10 の RANDOMIZE により RUN するごとに乱数の系列を変えることができます．

3.2.26 ERROR （エラー：error）

〔機　能〕

BASICのエラー発生をシミュレートしたり，ユーザのエラー番号の定義を可能にします．

〔形　式〕

ERROR　エラー番号

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・エラー番号は整数形式の数値式であり，その値は1～255でなければなりません．

・ERROR文が実行されると，ON ERROR GOTO文がある場合以外は，そのエラー番号に対応するエラーメッセージを出力して実行が停止します．もしエラー番号に対応するエラーコードがない場合には，"Unprintable Error"のメッセージが出力されます．

・ON ERROR GOTO文がある場合は，メッセージを出力しないで指定された行番号に分岐します．

・以上どちらの場合も，ERROR文が実行されるとERR変数とERL変数には，そのエラー番号とERROR文の行番号が代入されます．

・BASICのエラーコードにない番号を用いることにより，ユーザプログラムのエラーコードを定義することができます．

```
10 ON ERROR GOTO 80
20 RANDOMIZE (TIME)
30 A=INT(RND(1)*9)+1
40 INPUT B
50 IF A<B THEN ERROR 80 ELSE IF A>B THEN ERROR 81
60 PRINT "Solution !!"
70 END
80 IF ERR=80 THEN PRINT "Too Large !" ELSE PRINT "Too Small !"
90 RESUME 40
100 END

Ready
RUN
? 5
Too Large !
? 3
Solution !!

Ready
```

このプログラムは，数当てゲームになっています．正解でない場合エラートラップ機能により，エラー処理を行います．エラー処理ルーチンではERROR文で定義したエラー番号によりエラーの原因を判定することができます．

3.2.27 ON ERROR GOTO (オン・エラー・ゴーツー：on～error～goto)

〔機　能〕

エラートラップ機能を可能にします．

〔形　式〕

ON ERROR GOTO　行番号

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・行番号は，エラー処理ルーチンの最初の行を指定します．

・エラートラップ機能が可能になると，エラーが検出されたとき，エラーメッセージを出力せずに，指定されたエラー処理ルーチンに制御が移ります．

・エラーが起ったときには，自動的にそのエラー番号と行番号が変数ERRとERLにセーブされます．

・エラー処理ルーチンの中では，エラートラップ機能は働かなくなり，もしエラー処理ルーチンの中でエラーが起ったときはエラーメッセージが出力され，実行は停止します．

・エラートラップ機能を無効にするには，ON ERROR GOTO 0を実行します．

〔例〕

3.2.26 ERROR文を参照して下さい．

3.2.28 RESUME (リジューム：resume)

〔機 能〕

エラー処理終了後，プログラムの実行を再開します．

〔形 式〕

```
RESUME {NEXT | 行番号}
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・行番号を指定したときは，その行番号から実行を再開します．

・NEXT を指定したときは，エラーの起きた文の，次の文から実行を再開します．

・指定のないときは，エラーの起きた文から実行を再開します．
なお，このとき，エラー処理ルーチンでエラーの原因を取除かないと，またエラー処理ルーチンへ戻ります．

〔例〕

3.2.26 ERROR 文を参照して下さい．

3.2.29 BEEP （ビープ：beep）

〔機　能〕

内蔵スピーカによりブザーを鳴らします．

〔形　式〕

```
BEEP [スイッチ]
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・スイッチを1にすると，次にスイッチ0のBEEP文が実行されるまでブザーが鳴り続きます．

・スイッチを省略すると，PRINT CHR$(7)を実行したときと同じように，一定時間ブザーが鳴ります．

〔例〕

```
10 FOR I=1 TO 100
20 BEEP 0
30 FOR J=1 TO I
40 NEXT J
50 BEEP 1
60 FOR K=1 TO 15
70 NEXT K
80 NEXT I
90 BEEP 0
100 END
```

FOR～NEXTループで，ブザーの鳴っている時間を変えています．

3.2.30 MOTOR （モーター：motor）

〔機　能〕

カセットテープレコーダのモータを制御します．

〔形　式〕

MOTOR ［スイッチ］

（省略形は　M.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・スイッチをONにすればモータはオンになり，OFFにすればオフの状態になります．
・スイッチの指定がない場合，モータがONの状態ならオフに，OFFの状態ならオンになります．

〔例〕

MOTOR ON

カセットテープのモータをオン状態にします．

3.2.31 TRON （トレース・オン：trace on）

〔機　能〕

プログラムの実行状態を追跡します．

〔形　式〕

```
TRON
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・TRONを実行するとBASICはトレースモードに入ります．トレースモードでは，実行する行番号を表示してから，その行を実行します．

・TRONを一度実行すると，TROFFを実行するかNEWを行うまでトレースモードになっています．

〔例〕

```
10 TRON
20 I=1
30 PRINT I
40 I=I*2
50 IF I>=1024 THEN TROFF:END
60 GOTO 30
70 END
```

実行を行う行番号は〔行番号〕で表示されます．番号30のPRINT Iがありませんと，トレース行の表示はさらに右へ続きます．

```
[20][30] 1
[40][50][60][30] 2
[40][50][60][30] 4
[40][50][60][30] 8
[40][50][60][30] 16
[40][50][60][30] 32
[40][50][60][30] 64
[40][50][60][30] 128
[40][50][60][30] 256
[40][50][60][30] 512
[40][50]
Ready
```

3.2.32 TROFF（トレース・オフ：trace off）

〔機 能〕

プログラムの実行状態の追跡を止めます．

〔形 式〕

```
TROFF
```

〔バージョン〕 V1.0 V2.0 V3.0

〔例〕

3.2.31 TRON 文を参照して下さい．

3.2.33 CHAIN （チェイン：chain）

〔機　能〕

指定されたファイルのプログラムを呼び出し実行します．このとき，現在メモリ上にあるプログラムから変数を引き渡すことができます．

〔形　式〕

CHAIN [MERGE] "ファイルディスクリプタ"
[, [行番号式] [, ALL] [, DELETE 範囲]]

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・ファイルディスクリプタにより呼び出すプログラムを指定します．

・行番号式により，呼び出されたプログラムの実行開始行を指定します．行番号式が省略されたときは，プログラムの最初の行から実行されます．この行番号式は，単に一つの行番号であってもよいし，また式であってもかまいません．式のときは，その評価結果が実行開始行番号になります．行番号式に現われた行番号は，RENUM コマンドを実行しても変わりません．

・ALL の指定をすると，現在の全ての変数，配列が引き渡されます．この指定を省略すると，COMMON 文で指定された変数，配列だけが引き渡されます．したがって，必要な変数，配列だけを渡したいときは，ALL の指定をせずに，COMMON 文を用います．

・MERGE を指定すると，メモリ上のプログラムと呼び出されたプログラムとの混ぜ合わせが行われます．その結果，指定された行番号から実行が開始されます．
MERGE が指定されたとき，呼び出されるプログラムは，アスキー形式だけでなくバイナリ形式でもかまいません．ただしバイナリ形式ファイルの場合には，そのファイルの先頭の行番号は，メモリ上にあるプログラムの最大の行番号より大きくなければなりません．そうでないとき，"Bad File Mode" のエラーになります．
MERGE を指定すると，現在のプログラムで使用しているファイルは，CHAIN 実行後もオープン状態に保たれます．

・DELETE の指定は，MERGE の指定がある場合にのみ意味をもちます．MERGE の処理に先立って，範囲で指定された行が削除されます．範囲の指定のしかたは，DELETE コマンドと全く同じです．
この DELETE で指定される行番号は，RENUM コマンドにより変更することができます．

・MERGE の指定を省略すると，それまでのプログラムで行っていた変数の型の指定

や使用者関数の定義は無効になります．したがって，この様な場合，DEFINT, DEF SNG, DEFDBL, DEFSTR 及び DEFFN の各文は，CHAIN によって連結されたプログラムの中で，必要に応じて再実行されるようにしなければなりません．

〔例〕

メインプログラム

```
10 A=123
20 B=456
30 C$="F-BASIC"
40 CHAIN "0:SUB",,ALL
50 END
```

メインプログラムで，変数A，B，C＄にそれぞれ値が代入され，それを引数にしてサブプログラムをロードし，実行するというプログラムです．

サブプログラム

```
10 PRINT "A=";A
20 PRINT "B=";B
30 PRINT "C$=";C$
40 END

Ready
RUN
A= 123
B= 456
C$=F-BASIC

Ready
```

3.2.34 COMMON (コモン：common)

〔機　能〕

CHAIN 文により連結されるプログラムに変数を引き渡します.

〔形　式〕

COMMON　変数名［，変数名］…

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・COMMON 文は，連結されるプログラムに引き渡される変数を指定します．したがって，COMMON 文は必ず CHAIN 文と対で使用されます.
COMMON 文は非実行文ですから，プログラム中のどこにあってもかまいませんが，プログラムの先頭にある方が望ましいです.

・１つ又は複数の COMMON 文において，同じ変数を重複して使用してはなりません.
配列変数を指定する場合には，変数名の後に"(　)"を付与します.

・CHAIN 文により連結されるプログラムに全ての変数を引き渡すときには，COMMON 文は使わないで，CHAIN 文で ALL を指定する方が便利です.

・COMMON 文は，対応する CHAIN 文に ALL の指定が無い時にのみ意味を持ちます.
指定された配列名が未定義の場合には，"Illegal Function Call" のエラーになります.
エラーが起きたときには，CLEAR 文が実行されたのと同じ状態になり変数，配列は初期設定されます.

〔例〕

メインプログラム

```
10 A=123
20 B=456
30 C$="F-BASIC"
40 COMMON A,C$
50 CHAIN "0:SUB"
60 END
```

サブプログラム

```
10 PRINT "A=";A
20 PRINT "B=";B
30 PRINT "C$=";C$
40 END
```

```
Ready
RUN
A= 123
B= 0
C$=F-BASIC

Ready
```

3.2.35 ERASE （イレーズ：erase）

〔機　能〕

配列変数をプログラムから消去します.

〔形　式〕

ERASE　配列変数名［，配列変数名］…

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・配列変数を消去すると，その後，同一変数名の配列を DIM 文で新たに宣言することができます．もし ERASE 文によって消去せずに，同一変数名で DIM 文を用いると，"Duplicate Defin ition" のエラーになります．

・配列変数を消去すると，その配列変数に割り当てられていたメモリ領域は解放され，他の目的に使用することができます．

〔例〕

```
10 DIM A(10)
20 DIM B(10,20)
30 ERASE A
40 DIM A(30)
50 DIM B(10)

Ready
RUN

Duplicate Definition In 50
Ready
```

行番号 30 の ERASE 文で，配列 A は消去されているので 40 行で新たに配列宣言できますが，B は消去されていませんので，50 行でエラーになります．

3.3 入出力ステートメント

3.3.1 INPUT (インプット：input)

〔機　能〕

キーボードから入力されるデータを読取ります.

〔形　式〕

INPUT ["プロンプトメッセージ" {; | ,}]変数名[,変数名]…

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・INPUTを実行してデータの入力待ち状態になると，疑問符（？）が表示され，そこからデータの入力が可能になります.
・プロンプトメッセージを指定すると，疑問符の前にそのメッセージが出力されます.なお，プロンプトメッセージの後がコンマの場合は，疑問符は出力されません.
・入力するデータの型と対応する変数名の型および入力するデータの個数と変数名の個数は一致しなければなりません．もし型が一致しなかったり個数が同じでなかった場合は，"? Redo From Start" と出力して再度入力待ちとなります.
・データを複数個入力するときは，データの間をコンマまたはコロンで区切ります.
・文字定数を入力するときは引用符で囲む必要はありませんが，コンマ，コロンや文字列の前後の空白もデータとして入力する場合は，文字列全体を引用符で囲まなければなりません.
・CTRL－C，CTRL－XまたはBREAKキー（V1.0 V2.0ではSTOPキー）の入力により，BASICはプログラムの実行を中断してコマンドレベルに戻ります．次にCONTコマンドを入力すると，INPUT文で中断した場合，そのINPUT文から実行を再開することができます.

〔例〕

```
10 S=0
20 INPUT "ﾃﾞｰﾀ ﾉ ｺｽｳﾊ ";I
30 FOR J=1 TO I
40 INPUT "ﾃﾞｰﾀ ﾊ ";DA
50 S=S+DA
60 NEXT J
70 H=S/I
80 PRINT "ﾍｲｷﾝ=";H
90 END

Ready
RUN
ﾃﾞｰﾀ ﾉ ｺｽｳﾊ ? 3
ﾃﾞｰﾀ ﾊ ? 1
ﾃﾞｰﾀ ﾊ ? 2
ﾃﾞｰﾀ ﾊ ? 3
ﾍｲｷﾝ= 2
```

3.3.2 LINE INPUT （ライン・インプット：line input）

〔機　能〕

1行全体の文字列（255文字以内）を区切ることなく，文字変数に読込みます．

〔形　式〕

LINE INPUT ［"プロンプトメッセージ"{；/，}］
　　文字変数

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・プロンプトメッセージを指定すると，入力データを受ける前に，そのメッセージが画面に出力されます．
・入力を促す疑問符（？）は出力されません．
・キーボードから入力された全ての文字データが指定された文字変数に代入されます．
・CTRL－C，CTRL－X または BREAK キー（V1.0 V2.0 では STOP キー）の入力により，LINE INPUT 文の実行を中断し，コマンドレベルに戻ります．このとき，CONT コマンドを入力すると，LINE INPUT 文の初めから実行が再開されます．

〔例〕

```
10 LINE INPUT"DATA ";A$
20 IF A$="END" OR A$="end" THEN 50
30 PRINT A$
40 GOTO 10
50 END

Ready
RUN
DATA 12.36
12.36
DATA "ABC",DEF,"GHI"
"ABC",DEF,"GHI"
DATA END
```

LINE INPUT 文ではINPUT 文での区切り記号も，データとして入力可能です．

3.3.3 PRINT （プリント：print）

〔機　能〕

画面に式の評価結果を出力します．

〔形　式〕

PRINT　[式[{，/；}[式]]……]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式または文字式で数値式の場合は，その結果の値が出力され，文字式の場合は，その結果の文字列が出力されます．

・式を複数個書くときは，その間をコンマ（，），セミコロン（；）または空白で区切らなければいけません．空白は，セミコロンと同じ意味を持ちます．
ただし，変数と文字定数，および文字定数と文字定数との区切りに限っては区切り記号を省略することができます．このとき，セミコロンを指定したときと同等の効果があります．

・出力の形式は，式の区切り方により決められます．
セミコロンあるいは空白で区切ると，前の式の最後の値に続いて出力されます．数値の場合は前後に1個の空白がとられ，その値が負のときは，前の空白の所にマイナス（－）符号が入れられます．
コンマで区切ると，BASICは1行を14文字ずつのフィールドに分け，式の値はそのフィールドの初めから出力されます．式の結果が2つ以上のフィールドにまたがるときは，次の値は前の値の最後のフィールドの次から出力されます．式の値が数値のときは，先頭の1バイトは符号を出力するための領域ですが，正の値のときは空白が出力されます．

・式の最後がコンマまたはセミコロンで終わっている場合は，次のPRINT文による出力は，同じ行に上に述べた形式で出力されます．式の最後がコンマまたはセミコロンのいずれかで終わっていない場合は，改行されます．

・出力される行の長さが画面の幅よりも長いときは，次の行に続けて出力されます．また現在の行において，カーソル位置から行の終わりまでの間に次の文字列の長さ，数値の長さを確保できない場合には改行して表示されます．

・式が1つもないときは，単に改行のみが行われます．

・キーワードPRINTの代わりに疑問符（？）を書くことができます．

〔例〕

```
10 A$="PRINT"
20 B=1234:C$="F-BASIC":D$="ﾍﾞｰｼｯｸ"
30 PRINT A$
40 PRINT
50 PRINT C$,D$
60 PRINT B,C$;D$
70 PRINT A$;"END"
80 END

Ready
RUN
PRINT

F-BASIC       ﾍﾞｰｼｯｸ
 1234         F-BASICﾍﾞｰｼｯｸ
PRINTEND
```

3.3.4 LPRINT (エル・プリント：line print out)

〔機 能〕

プリンタに式の評価結果を出力します。

〔形 式〕

```
LPRINT [式[{;/,} [式]] …]
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・データの出力先がプリンタであるという点を除いて，使い方はPRINT文と全く同じです.

〔例〕

```
10 FOR I=&H20 TO &H3F
20 LPRINT CHR$(I);
30 NEXT
```

```
 !"#$%&'()*+,-./0123456789:;<=>?
```

3.3.5 PRINT@ （プリント・アットマーク：print @）

〔機　能〕

漢字を画面に表示します．

〔形　式〕

PRINT@　[(*x*, *y*),]漢字コード[{ , ; } [漢字コード]]…

〔バージョン〕　(V1.0)　(V2.0)　(V3.0)

〔説　明〕

- 漢字コード（JIS漢字コード系に従う）により示される漢字列を画面に表示します．
- (*x*, *y*) は，漢字コード列の先頭の漢字を表示するグラフィック座標を示します．
- 漢字コードの最後がコンマまたはセミコロンで終わっているときは，次のPRINT @文で示される漢字コードは，同じ行に続けて表示されます．また，漢字コード並びの最後にコンマまたはセミコロンがないときは，次のPRINT @文で示される漢字列は，次の行の先頭から表示されます．
  ただし，次のPRINT@文で座標 (*x*, *y*) が書かれているときは，その座標が優先されます．
- 1つの漢字は16×16のドットパターンにより構成されるため，1行には最大40字の漢字が，画面には最大12行の表示ができます．

〔例〕

```
10 CLS
20 PRINT@(100,130) ,&H3441,&H3B7A,&H244F,&H2350,&H2352
   ,&H2349,&H234E,&H2354,&H2177,&H4A38,&H2447,&H2122
30 PRINT@(100,150) ,&H3268,&H4C4C,&H244B,&H3D50,&H4E4F
   ,&H2447,&H242D,&H245E,&H2439,&H2123
```

漢字はＰＲＩＮＴ＠文で、
画面に出力できます。

3.3.6 PRINT USING（プリント・ユーズィング：print using）

〔機　能〕

文字または数値を指定した書式で画面に出力します．

〔形　式〕

PRINT USING フォーマット文字列;出力ならび

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

出力ならびは，セミコロンまたはコンマで区切られた文字式，または数値式です．フォーマット文字列は文字式で，出力される文字や数値の書式を決定します．フォーマット文字列の中で書ける書式制御文字には次のようなものがありますが，これらの書式制御文字はそのまま文字として画面に出力されません．

文字表示の書式制御文字

（1）！（エクスクラメーションマーク）

与えられた文字列の最初の一文字だけを出力します．

（2）& *n* 個の空白&

n 個の空白（ $n \geqq 0$ ）を2つの〝&〟で囲みます．このとき，与えられた文字列の先頭から $n+2$ 個の文字が出力され，残りの文字は無視されます．また，指定した長さ（ $n+2$ ）よりも文字列の方が短いときは，文字は文字領域の左からつめられ，残りの部分には空白が出力されます．

（3）@（アットマーク）……V2.0 V3.0 で有効

可変長の文字領域を定義します．〝@〟が指定されると，出力ならびの中の対応する文字列がそのまま出力されます．

数値表示の書式制御文字

（1）＃（ナンバ記号）

ナンバ記号の個数によって出力する数字の桁数を指定します．数値は右づめで出力され，左側のあいた部分には空白がつめられます．

負の数値の場合には負符号が先頭の数字の左に出力され，この符号も1桁として数えられ1個の＃が使用されます．

（2）．(小数点)

数値領域の任意の個所に小数点を挿入することを指示します．小数点に続いてナンバ記号がある場合は，小数点以下に指定された桁数分は必ず出力されます．

(3) ＋（プラス）

数値書式制御文字の最初または最後のプラス記号は，その数値の前または後にその数値の符号（プラスまたはマイナス）を付けることを指示します．

(4) －（マイナス）

数値書式制御文字の最後のマイナス記号は，負の数値の後にマイナス符号を付けることを指示します．

(5) ＊＊（2個のアスタリスク）

数値書式制御文字の最初に置かれます．このとき，数値領域の上位桁の空白の部分にアスタリスクが出力されます．2個のアスタリスクは，2桁分の領域を確保します．

(6) ¥¥（2個の円記号）

数値書式制御文字の最初に2つの円記号を書くと，出力される数字の直前の空白の位置に円記号を出力します．2つの円記号により2桁分の領域が確保されますが，円記号はこのうち1個だけを使用します．

(7) ＊＊¥（2個のアスタリスクと円記号）

数値書式制御文字の最初に＊＊¥を書くと，(5)と(6)の両方の機能を果します．ただし，＊＊¥は3桁分の領域が確保されますが，そのうちの1個は円記号に使用します．

(8) ,（コンマ）

小数点位置指定の〝.〟の左側に置かれます．このとき，整数部分を右側から3桁ごとにコンマで区切ります．コンマを表示するたびにそのための桁が確保されます．

(9) ^^^^（4個の矢印）

桁指定文字のナンバ記号の後に置かれます．この指定により指数形式で出力することができます．4個の矢印は，E±*nn*又はD±*nn*が出力される領域を確保します．ナンバ記号により有効数字が指定され，指数はそれに合わせて調整されます．

(10) _（アンダースコア記号）……(V2.0)(V3.0)で有効

アンダースコア記号に続く1文字を書式制御の意味をもたない文字としてそのまま出力します．アンダースコア記号自身を文字として出力するためには，アンダースコア記号を2つ並べて用います．

文字列の表示

フォーマット文字列の中に，書式制御文字以外の文字を書くことができます．このとき，その文字は書式変換された文字の前後にそのまま画面に表示されます．

注1） 実際の数値が指定された桁数で表わせないときは，数値の先頭にパーセント記号(%)が出力されます．数値の丸めの結果，指定桁数をこえた場合でも，丸められた数値の前にパーセント記号が出力されます．

〔例〕

```
10 A=12345:B=62.35
20 C=-160
30 D=3.251E+12
40 PRINT USING "########  ######.#";A;A
50 PRINT USING "+#######  #######-";B;C
60 PRINT USING "**##.###  ¥¥######";B;C
70 PRINT USING "**¥#####  **¥####,";B;A
80 PRINT USING "#####^^^^";D

Ready
RUN
   12345   12345.0
     +62      160-
**62.350     -¥160
*****¥62  *¥12,345
 3251E+09

Ready
```

```
10 A=65537!
20 PRINT USING"#####_!";A

Ready
RUN
65537!

Ready
```

行番号20でのPRINT USING文中のアンダースコア記号により，エクスクラメーションは，書式制御文字としての意味を持ちません．

```
10 A$="BASIC":B$="F-BASIC":C$="ﾍﾞｰｼｯｸ"
20 PRINT USING "&     &   &         & !";A$;B$;C$
30 END

Ready
RUN
BASIC    F-BASIC  ﾍ

Ready
```

3.3.7 LPRINT USING （エル・プリント・ユーズィング：line print out using）

〔機 能〕

文字または数値を指定された書式でプリンタに出力します.

〔形 式〕

LPRINT USING フォーマット文字列；出力ならび

〔バージョン〕 (V1.0) (V2.0) (V3.0)

〔説 明〕

データの出力先がプリンタであるという点を除けば，使い方は PRINT USING と全く同じです.

```
10 A=123
20 B=567
30 C$="FUJITSU"
40 LPRINT USING "##### ###.####";A;B
50 LPRINT USING "@";C$
60 END
```

```
  123 567.0000
FUJITSU
```

3.3.8 OPEN (オープン：open)

〔機　能〕

ファイルのオープン処理を行います．

〔形　式〕

```
OPEN 〝モード〞, [#]ファイル番号,
     〝ファイルディスクリプタ〞
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ファイルディスクリプタで示されるファイルにファイル番号と入出力バッファを割当て，以下そのファイル番号での入出力を可能にします．

・INPUT # 文およびPRINT # 文，LINE INPUT # 文，GET 文，PUT 文及びファイル番号を必要とする関数によりファイルからのデータの入出力を行うときは，最初に必ずOPEN 文を実行しなければなりません．

・モードはモード文字，I，O，A またはRにより，そのファイルの入出力モードを指定します．

I　指定されたファイルからの入力処理を行います．指定されたファイル名は，指定されたデバイスの媒体上に存在しなければなりません．

O　指定されたファイルへの出力処理を行います．指定ファイル名は，指定されたデバイスの媒体上に存在していてはなりません．

A　シーケンシャルファイルにデータの追加出力を行います．指定されたファイル名は，ディスク上に存在しなければなりません．この指定はディスク上のファイルに対してのみ有効です．

R　ランダムファイルの入出力処理を行います．この指定はディスク上のファイルに対してのみ有効です．指定されたファイル名がディスク上に存在しないときは，そのファイルが新たに登録されます．

・ファイル番号は1から16でなければなりません．

・ディスク上のファイルの場合は，複数個のファイル番号で同一のファイルをオープンすることができます．このとき，その入出力モードはIまたはRでなければなりません．

・デバイス名がバブルカセットまたはオーディオカセットの場合は，同一ユニットで同時にオープンできるファイルは1個だけです．

・RS−232C およびプリンタのファイルディスクリプタの記述のとき，オプションの指定ができます．プリンタの場合には，S またはWによりプリンタの桁数を指定できま

す．Sの指定をすると，プリンタの印字桁数を80桁とする制御を行い，Wの指定をすると，136桁とする制御を行います．この指定は一度行うと，次の指定があるまで有効です．なお，BASICの起動時にはSに初期設定されています．RS－232Cのオプションについては，「3.8.3　OPEN」を参照して下さい．

〔例〕

```
10 OPEN "O",#1,"O:DATA"
20 READ A$
30 IF A$="E" THEN 90
40 READ B
50 PRINT #1,A$
60 PRINT #1,B
70 GOTO 20
80 DATA IC,200,ﾊﾟｿｺﾝ,300,ﾏｲｺﾝ,6809,E
90 CLOSE #1
100 STOP
110 PRINT "ｳﾘｱｹﾞﾀﾞｶ"
120 OPEN "I",#2,"O:DATA"
130 FOR I=1 TO 3
140 INPUT #2,A$
150 INPUT #2,B
160 PRINT A$,B
170 NEXT I
180 END
```

行番号10から90までで，フロッピィディスクにシーケンシャイファイルを作成します．行番号100のSTOP文で実行が止まりますので，CONTコマンドを入力すると実行を再開し，行番号110から170でシーケンシャルファイルからデータを読み画面に出力します．

3.3.9 CLOSE （クローズ：close）

〔機　能〕

ファイルのクローズ処理を行います.

〔形　式〕

```
CLOSE  [[#]ファイル番号
       [,[#]ファイル番号]…]
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ファイル番号とファイルディスクリプタの割当てを解除し，その入出力バッファを他のファイルで使用できるようにします.

・ファイル番号を必要とするステートメントによる入出力処理の終わりには，必ずCLOSE文を実行しなければなりません.

・ファイル番号の指定のないCLOSE文を実行すると，オープンされている全てのファイルがクローズされます.

・END文を実行すると，自動的にすべてのファイルがクローズされます. ただし，STOP文はクローズ処理を行いません.

〔例〕

3.3.8 OPEN文を参照して下さい.

3.3.10 INPUT＃ （インプット・シャープ：input ＃）

〔機 能〕

ファイルからデータを読込み，変数に代入します．

〔形 式〕

INPUT＃ ファイル番号，変数名[,変数名]…

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・ファイル番号は，OPEN 文によりファイルを入力モード（"I"）でオープンした番号を指定します．

・変数名は，読込んだデータを格納する領域で，読込むデータの型と変数の型は一致してなければなりません．データを文字変数に読込むとき，コンマ，コロン，CRまたはLF がデータの区切りになります．データの前後の空白は無視されます．したがって，これらの文字をデータとして入力するならば，引用符(")で囲った文字列として出力されていなければなりません．ただし引用符で囲まれた文字列は，引用符を文字として含むことはできません．データを数値変数に読込むときは，コンマ，コロン，空白，CR，LF がデータの区切りになります．データに先行する空白は無視されます．

〔例〕

3.3.8 OPEN 文を参照して下さい．

3.3.11 PRINT＃ （プリント・シャープ：print ＃）

〔機 能〕

式の評価結果を指定されたファイルに出力します．

〔形 式〕

PRINT＃ ファイル番号[,出力ならび]

または

PRINT＃ ファイル番号, USING フォーマット文字列 ;出力ならび

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・ファイル番号は，OPEN 文によってそのファイルを出力モード（"O"）または追加モード（"A"）でオープンしたときの番号です．

・出力ならびは，その値がファイルに書込まれる数値式または文字式です．式が2つ以上あるときは，その間をセミコロンまたはコンマで区切ります．

・フロッピィディスクに文字データを出力するときは，その直後に区切り記号としてコンマ(,)を出力させます．もし，コンマを出力しないと，文字データを続いて出力したとき，それらのデータは連続した文字列として出力されるため，INPUT 文では1つの文字データとして読込まれてしまいます．バブルカセットおよびオーディオカセットに出力するときは，1つのデータの後に必ずCRコードが出力されるため，上記の区切り記号を出力する必要はありません．

・引用符を書込もうとするときは，CHR＄（34）を用います．

〔例〕

```
10 OPEN "O",#1,"LPTO:"
20 INPUT "ﾅﾏｴ ﾊ ";A$
30 PRINT #1,A$;
40 INPUT "TEL ﾊ ";B$
50 PRINT #1,"      ";B$
60 CLOSE #1
```

キーボードから入力された文字を，プリンタに出力します．

3.3.12 LINE INPUT＃ （ライン・インプット・シャープ：line input ＃）

〔機　能〕

指定されたファイルから1行を区切ることなく読込み，変数に代入します．

〔形　式〕

```
LINE INPUT＃　ファイル番号，文字変数
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイル番号は，OPEN文によってそのファイルを入力モード（"I"）で，オープンしたときに指定した番号です．

・文字変数は，1行全体が代入される文字変数名です．

・LINE INPUT＃文は，CRコードまでの全ての文字を区切ることなく，指定された文字変数に読込みます．読込める文字数の最大値は255までです．
なお，CRコードまでの間にCRコード以外のコントロールコードがある場合は，そのコードは読込まれません．

〔例〕

```
10 OPEN"O",#1,"0:DATA"
20 READ A$
30 IF A$="E" THEN 80
40 READ B
50 PRINT#1,A$,B
60 GOTO 20
70 DATA ﾀﾛｰ,100,ﾊﾅｺ,101,ｼﾞﾛｰ,102,E
80 CLOSE#1
90 STOP
100 PRINT"ｼﾒｲ                NO."
110 PRINT
120 OPEN"I",#2,"0:DATA"
130 LINE INPUT#2,A$
140 PRINT A$
150 IF EOF(2) THEN 170
160 GOTO 130
170 CLOSE#2
180 END
```

```
RUN

Break In 90
Ready
CONT
ｼﾒｲ                NO.

ﾀﾛｰ                100
ﾊﾅｺ                101
ｼﾞﾛｰ               102

Ready
```

3.3.13 FIELD （フィールド：field） （ディスク）

〔機　能〕

ランダムファイルのバッファに変数の領域を割当てます．

〔形　式〕

FIELD ［#］ファイル番号，フィールド幅　AS 文字変数名［,フィールド幅　AS 文字変数名］…

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

- GET 文又は PUT 文によりランダムファイルの入出力を行うときは，FIELD 文によりランダムファイルバッファを各変数のデータを貯えるフィールドに分割しなければなりません．バッファの大きさは 256 バイトなので，各文字変数のフィールド幅を加えた値が 256 以内でなければなりません．
- ファイル番号は，そのファイルをランダム入出力モード（"R"）でオープンしたときに指定した番号です．
- 特にファイル番号の 0 は，DSKI$，DSKO$ において使用されるシステムランダムバッファを定義するときに用います．
- フィールド幅は，文字変数に割当てる文字数です．
- FIELD 文の文字変数の定義は，必ず LSET/RSET で行わなければなりません．また，代入文またはINPUT文等により，FIELD 文で使われた文字変数の定義を行ってはなりません．
- 同じファイル番号に対して複数個の FIELD 文を実行することができます．そのとき，各々の FIELD 文はバッファの先頭位置からバッファを再定義します．したがって，この様にすると同一のフィールドを違った変数名で参照することができます．

〔例〕

```
10 OPEN "R",#1,"DATA"
20 FIELD #1,20 AS A$,30 AS B$,40 AS C$
30 FIELD #1,90 AS DUMMY$,4 AS D$,4 AS E$
40 GET #1
```

DUMMY$ の内容は，A$＋B$＋C$ と同じになります．

3.3.14 GET （ゲット：get）

（ディスク）

〔機　能〕

ランダムファイルから指定されたレコードをバッファに読込みます．

〔形　式〕

```
GET 〔#〕 ファイル番号 〔,レコード番号〕
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ファイル番号は，そのファイルをランダム入出力モード（"R"）でオープンしたときに指定した番号です．

・レコード番号は，ランダムファイルの何番目のレコードを読込むかを示し，その値は1から最大レコード番号までで，読込まれるレコードの大きさは256バイトです．
レコード番号の指定のないときは，直前にそのファイルに対してGETまたは，PUTしたレコードの次のレコードが読込まれます．

〔例〕

```
10 OPEN "R",#1,"DATA1"
20 FIELD #1,20 AS NAM$,20 AS TEL$
30 INPUT "ﾚｺｰﾄﾞ ﾊﾞﾝｺﾞｳﾊ ";A
40 IF A>100 THEN 110
50 GET #1,A
60 A$=NAM$
70 B$=TEL$
80 PRINT A$,B$
90 PRINT
100 GO TO 30
110 CLOSE #1
120 END
```

フロッピィディスクのDATA 1というランダムファイルからデータを読んで画面に出力します．

3.3.15 PUT (プット：put)　　（ディスク）

〔機　能〕

バッファの内容を指定されたランダムファイルに書込みます．

〔形　式〕

PUT　[#]ファイル番号[,レコード番号]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイル番号は，OPEN文によりそのファイルをランダム入出力モード（"R"）で，オープンしたとき指定した番号です．

・レコード番号は，ランダムファイルの何番目のレコードに書込むかを示します．その値は1からnnnまで指定できます．

・レコード番号が省略されたときは，直前にそのファイルに対してPUTまたは，GETしたレコードの次のレコードに出力されます．

〔例〕

```
10 OPEN "R",#1,"DATA1"
20 FIELD #1,20 AS NAM$,20 AS TEL$
30 INPUT "ﾅﾏｴﾊ ";A$
40 IF A$="END" THEN 100
50 INPUT"TEL ";B$
60 LSET NAM$=A$
70 LSET TEL$=B$
80 PUT #1
90 GOTO 30
100 CLOSE #1
110 END
```

キーボードより入力されたデータを，DATA1というランダムファイルに書込みます．

3.3.16 DSKO$ （ディスク・オー・ダラー：disk output $） （ディスク）

〔機　能〕

システムランダムバッファの内容を指定されたセクタに書込みます.

〔形　式〕

DSKO$　ドライブ番号, トラック番号, セクタ番号

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・この命令は, 指定したセクタに直接, データを書込みますのでディスクファイルを壊す恐れがあります.
使用の際は, 十分注意して下さい.

・このコマンドを使用する前にFIELD文により, システムランダムバッファに割当てる文字変数を定義しておかなければなりません.

・ドライブ番号は, システムの構成により0から3, または0から7まで指定できます.

・トラック番号は, ミニフロッピィディスクのときは0から39まで, 標準フロッピィディスクのときは0から76まで指定できます.

・セクタ番号は, ミニフロッピィディスクのときは1から32まで指定できます.
17～32を指定したときは, ディスクの裏面のセクタ1～16を示します.
また, 標準フロッピィディスクのときは1から52まで指定できます.
27～52を指定したときは, ディスクの裏面のセクタ1～26を示します.

・システムランダムバッファはシステムに持つ256バイト領域のことです.

〔例〕

```
10 FIELD #0,128 AS A$,128 AS B$
20 DUMMY$=DSKI$(0,10,1)
30 DSKO$ 0,39,1
```

ドライブ0のトラック10, セクタ1の内容を読込み, トラック39, セクタ1に書込みます.

3.3.17 BUBW (バブ・ライト：bubble write)

〔機　能〕

変数または配列の内容をバブルカセットの指定されたページに書込みます.

〔形　式〕

BUBW　ユニット番号, ページ番号, {文字変数名 / 配列名}

〔バージョン〕 (V1.0) (V2.0) (V3.0)

〔説　明〕

・ユニット番号は0または1が指定できます. ページ番号は0から1023まで指定でき, 1ページの大きさは32バイトです.

・書込むデータが文字列のときは, そのデータが代入されている文字変数名を指定します. また, 書込むデータが数値のときは, その数値が代入されている配列名を指定します.

・指定した配列の大きさまたは文字変数の文字数が32バイトでないときは, 次のように処理されます.

(1) 32バイトに満たないとき,
文字データの場合は, 残りの部分に空白がつめられ, 数値データの場合は0がつめられます.

(2) 32バイトを越えるとき,
文字データの場合は, 越えた部分は書込まれません. 数値データの場合は, 連続するページに次々と全てのデータが書込まれます. このとき, 最後のページに書くデータが32バイトに満たないときは(1)と同様の処理がされます.

〔例〕

```
10 A$="FUJITSU"
20 B$="PERSONAL"
30 C$="COMPUTER"
40 X$="A$+B$+C$
50 BUBW 0,512,X$
```

・この命令でデータを書込んだ場合は, F-BASICの管理外になるため, 使用状態をユーザ自身で管理しなければなりません. 特に0ページには, ボリュームラベル(先頭に"VOL 00000"が書かれている)が記録されているので, この情報をBUBW文により書換えた場合には, 以後, FILES文を実行すると"Device Unavailable"エラーになりますので注意して下さい.

3.3.18 BUBR (バブ・リード：bubble read)

〔機　能〕

バブルカセットの指定されたページの内容を変数または配列に読込みます．

〔形　式〕

BUBR　ユニット番号，ページ番号，{文字変数名 / 配列名}

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ユニット番号は0または1が指定できます．ページ番号は0から1023まで指定できます．
・文字変数名が指定されたときは，指定されたページの内容がそのまま変数に代入されます．
・配列名が指定されたときは，ページ番号を読込み開始番号として，配列の大きさに相当するページ数が読込まれ，配列にそのまま代入されます．

〔例〕

```
10 A$="FUJITSU "
20 B$="PERSONAL "
30 C$="COMPUTER"
40 X$=A$+B$+C$
50 BUBW 0,512,X$
60 BUBR 0,512,Y$
70 PRINT Y$

RUN
FUJITSU PERSONAL COMPUTER
```

3.4 画面制御・グラフィック機能

3.4.1 WIDTH （ウィドス：width）

〔機　能〕

画面に表示する文字の行数と桁数を指定します．

〔形　式〕

WIDTH　［1行の文字数］［,1画面の行数］

（省略形は　W.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・1行の文字数として，80（字），40（字）の指定ができます．
・1画面の行数として，25（行），20（行）の指定ができます．
・このステートメントを実行すると，画面が同時にクリアされます．
・BASICが起動されたときは，40（字）×20（行）に初期設定されています．
・WIDTH　,20を実行したときは，同時にCONSOLE　0，20，0，0（3.4.2を参照）が実行されます．

〔例〕

```
WIDTH 80,25
```

画面を80（字）×25（行）に設定します．

3.4.2 CONSOLE （コンソール：console）

〔機　能〕

スクロールウインドの大きさを指定します．

〔形　式〕

CONSOLE [スクロール開始行][,[スクロール行数] [,[ファンクションキー表示スイッチ] [,コンソールカラースイッチ]]]

（省略形は　CONS.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・スクロール開始行とスクロール行数によって，スクロールウインド，および，ページモード画面を設定します．
スクロール開始行は，0～24 の範囲で指定できます．
スクロールの行数は，0～25 の範囲で指定できます．
スクロール開始行を 0 または24（1画面20行のときは19），スクロール行数を 0 としたときは，全画面がページモードになります．
スクロール開始行を 1～23，スクロール行数を 0 としたときは，画面が二つのページ画面に分割され，上の画面をページモード 1 画面，下の画面をページモード 2 画面と呼びます．

・ファンクションキー表示スイッチを 1 にすれば，画面の下の 2 行にファンクションキーの内容が表示され，このときスクロールの対象となる行は 2 行少なくなります．
この表示スイッチを 0 にすれば表示しなくなります．

・コンソールカラースイッチは，画面に表示される文字を単色にするか，指定色にするかのスイッチです．
0 にすると，COLOR 文で指定されている色で表示されます．
1 にすると，グリーンのみで表示されます．この指定を行うと，文字の画面出力動作が速くなります．
ただし，単色モード指定のとき，カラーグラフィックを使用し，その後プログラムリストを出力すると，一部画像が残ったままになります．これを避けるためには，リスト出力前にクリアキーを入力するか，CLS 文を実行します．

・BASIC が起動されたときは，CONSOLE　0, 20, 0, 0 に初期設定されています．

〔例〕

```
CONSOLE 0,17,1,0
```

画面のスクロール範囲を0行から15行に設定し，ファンクションキーの内容を画面の下2行に表示します．

```
CONSOLE 10,5,0,0
```

一番上の行から9行目までをページモード1画面，10行目から14行目までをスクロール範囲，15行目から一番下の行までをページモード2画面に設定します．

画　面

3.4.3 COLOR（カラー：color）

〔形式1〕

COLOR　［カラーコード］［,背景色カラーコード］

（省略形は　COL.）

〔機　能〕

画面に表示する文字およびグラフィックの色，および背景色を指定します．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・カラーコードは表示する文字，およびグラフィックの色を示しています．

0(8)—黒	4(12)—緑
1(9)—青	5(13)—水色
2(10)—赤	6(14)—黄
3(11)—紫	7(15)—白

8～15の指定をしますと，現在の背景色が文字色となり，指定した色が背景色となります．なお，8～15の指定は，文字出力の場合にのみ有効です．

・背景色カラーコードは，画面の背景の色を示し，0～7で指定します．画面が指定した背景色になるのは，次にCLS文が実行されたときからです．

・BASICが起動されたときは，COLOR　7，0に初期設定されています．

〔例〕

```
10 WIDTH 80,25:CLS
20 FOR I=1 TO 1920
30 C=CINT(RND(1)*6)+1
40 COLOR C
50 PRINT "█";
60 NEXT I
70 LOCATE 0,0
80 COLOR 7,0
90 END
```

表示するカラーを乱数によって決めています．

〔形式2〕

```
COLOR [フォアグラウンドカラー][,バックグラウンドカラー]
```

〔機　能〕

画面の表示色及び背景色を指定します．

〔バージョン〕　(V1.0)　(V2.0)　(V3.0)

〔説　明〕

・フォアグラウンドカラーにより，これから表示する文字の色を指定します．このフォアグラウンドカラーは，パレットコードであり，カラーコードではありません．ここで指定されたパレットコードをどの色に対応づけるかは，形式3のCOLOR文で行います．

フォアグラウンドカラーの指定は，0～7のパレットコードで行いますが，パレットコードに8を加えた8～15の指定を行うと，文字の表示色と背景色が逆になったりバース表示がされます．

・バックグラウンドカラーにより，画面の背景の色を指定します．この命令を実行後，CLS文を実行すると画面が塗りかえられ，指定された背景色になります．

このバックグラウンドカラーは，フォアグラウンドカラーと同様に0～7のパレットコードにより行います．

・BASICが起動されたときは，COLOR 7，0に初期設定されています．

・グラフィックステートメントにおいて，そのパレットコードを省略したときは，直前に実行されたCOLOR文のフォアグラウンドカラーが採用されます．

〔例〕

〔形式3〕を参照して下さい．

〔形式 3〕

```
COLOR＝(パレットコード, カラーコード)
```

〔機　能〕

カラーパレットの定義を行います.

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・F-BASICの各グラフィックステートメントの色の指定は, すべてパレットコードにより行いますが, このCOLOR文は, パレットコードとカラーコードとの対応づけを行うものです。
BASICが起動されたときには, パレットコードとカラーコードは, 次の様に1対1に対応しています.

パレットコード		カラーコード	
0	――――	0	黒
1	――――	1	青
2	――――	2	赤
3	――――	3	紫
4	――――	4	緑
5	――――	5	水色
6	――――	6	黄
7	――――	7	白

したがって, このCOLOR文の機能を用いなければ, パレットコードはカラーコードと全く同じ様に使用できます.
このCOLOR文は, カラーパレットを任意の色に割り付けることができます. たとえば, 青で表示されているものを赤に, 赤で表示されているものを緑にといった具合に自由に色を変えることができます. このCOLOR文の実行後は, 現在そのパレットコードで画面に表示されている文字及びグラフィックスの色が指定された色に瞬時に変わります. また, 以後表示する文字及びグラフィックスもそのパレットコードに割り付けられたカラーで行われます.

・パレットコード0は, 黒以外の色に割り付けることができませんので, 注意して下さい.

〔例〕

```
10 WIDTH 80,25
20 CLS
30 COLOR=(7,1)
40 COLOR 0,7
45 CLS
46 LINE(220,50)-(420,150),PSET,1,BF
50 FOR I=1 TO 500
60 NEXT I
80 FOR I=2 TO 7
90 COLOR=(7,I)
100 LOCATE 0,0:PRINT "COLOR=( 7,";I;")"
110 FOR J=1 TO 1000
120 NEXT J
130 NEXT I
140 FOR I=1 TO 7
150 COLOR=(I,I)
160 NEXT
170 END
```

3.4.4 SCREEN （スクリーン：screen）

〔機　能〕

アクティブ画面とディスプレイ画面の設定します．

〔形　式〕

```
SCREEN [アクティブVRAMコード]
       [,ディスプレイ VRAMコード]
```

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・アクティブVRAMコードは，以後の画面上への動作に対して，B・R・GのどのVRAMが書込み可能かを指定するものです．この指定は0～7の範囲で行いますが，その値を2進数で表わしたときの3ビットの各ビットをB・R・Gに対応させ，ビットの0と1によりアクティブVRAMを表わします．すなわち，ビットが1のVRAMはアクティブになりデータの書込みが可能になります．ビットが0のVRAMはマスクされデータの書込みが不可能になります．(以前の状態が保存されます．)

アクティブ VRAMコード	ビット2 ビット1 ビット0 G R B	意　味
0	0 0 0	全てのVRAMがマスク
1	0 0 1	Bがアクティブ，RとGはマスク
2	0 1 0	Rがアクティブ，BとGはマスク
3	0 1 1	BとRがアクティブ，Gはマスク
4	1 0 0	Gがアクティブ，BとRはマスク
5	1 0 1	BとGがアクティブ，Rはマスク
6	1 1 0	RとGがアクティブ，Bはマスク
7	1 1 1	全てのVRAMがアクティブ

以上のようにアクティブVRAMコードにより，書込み可能画面が決まりますが，この機能を用いるときは，以後のステートメントにおけるパレットコードに注意しなければなりません．すなわち，現在指定されているパレットコードにより，B・R・GのどのVRAMに有効なデータを書くかが決められるため，そのVRAMがアクティブになってなければなりません．したがって，アクティブVRAMコードとパレットコードの論理積の結果が0であってはなりません．

・ディスプレイVRAMコードは，B・R・GのどのVRAMを画面に表示するかを指定します．0～7の値に対して，以下の意味を表わします．

ディスプレイVRAM コード	意　味
0	どのVRAMも表示しない
1	Bだけを表示
2	Rだけを表示
3	BとRを合成して表示
4	Gだけを表示
5	BとGを合成して表示
6	RとGを合成して表示
7	全てのVRAMを表示

・アクティブVRAMコード，ディスプレイVRAMコードを設定するときは，充分な注意が必要です．アクティブVRAMコードとパレットコードの設定が正しくなかったために VRAM に何も書かれていなかったり，ディスプレイ画面を間違えたために画面に何も表示されないことが起りえます．たとえば，COLOR 4：SCREEN 4，2 を実行すると，実行後BASICがコマンドレベルになっているにもかかわらず Ready の表示がされないで画面は真黒になっています．このような場合には，PF 9（SCREEN 7，7 が定義されている.）の入力により，アクティブ VRAM コードとディスプレイ VRAMコードを初期値に戻すようにしてください．

・BASICが起動したときは，SCREEN　7,7に初期設定されています．

〔例〕

```
10 WIDTH 80,25
20 CLS
30 SCREEN 7,0
40 FOR I=1 TO 7
50 COLOR I,0
60 PRINT STRING$(80,"■");
70 NEXT I
80 FOR I=0 TO 7
90 SCREEN 0,I
100 FOR J=1 TO 2000
110 NEXT J
120 NEXT I
130 SCREEN 7,7
140 END
```

3.4.5 CLS （シー・エル・エス：clear screen）

〔機　能〕

指定画面をクリアし，カーソルをその画面のホームポジションに移します．

〔形　式〕

CLS ［消去範囲コード］

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・消去範囲コードは，0～3を指定でき，次のことを示します．

0のとき，全画面をクリアします．

1のとき，スクロール画面をクリアします．

2のとき，ページモード1画面をクリアします．

3のとき，ページモード2画面をクリアします．

消去範囲コードが省略されたときは，0が指定されたものと見なされます．

〔例〕

```
10 CONSOLE 10,0,0,0
20 CLS 2
    .
    .
    .
80 CLS 3
```

CLS　2により，ページモード1画面をクリアし，カーソルをその画面のホームポジションに置きます．また，CLS　3により，ページモード2画面をクリアし，カーソルをその画面のホームポジションに置きます．

3.4.6 LOCATE （ロケート：locate）

〔機　能〕

CRT画面上の任意の位置にカーソルを移動します．

〔形　式〕

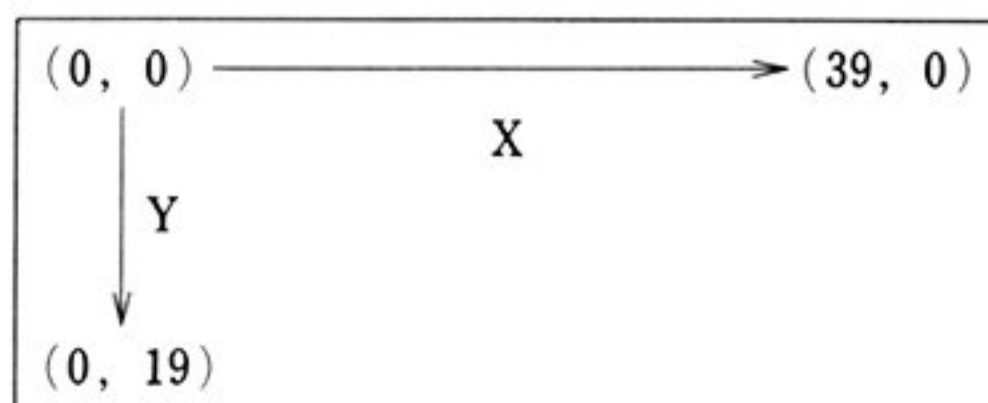
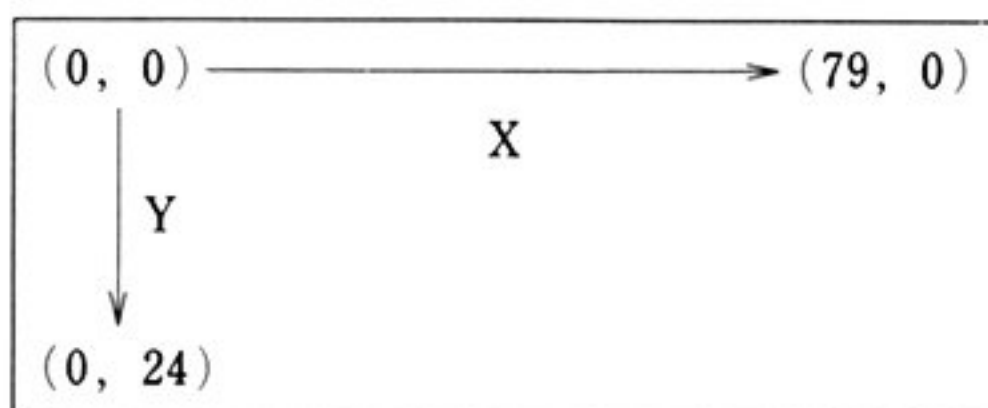
LOCATE　水平位置，垂直位置 [,カーソル表示スイッチ]

（省略形は　LOC.）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・水平位置，垂直位置は，カーソルを移動する座標を示します．
・カーソル表示スイッチを1にすると，指定位置にカーソルの表示を行い，0にすると，カーソルの表示を行いません．カーソル表示スイッチを省略したときは，直前に実行したLOCATE文のカーソル表示スイッチが有効になります（最初のLOCATE文では，カーソル表示スイッチは1に設定されています）．
・水平位置および垂直位置の指定できる範囲は，WIDTH文で指定された画面モードにより，次のようになります．

① WIDTH　40，20のとき

② WIDTH　80，25のとき

なお，画面上にファンクションキーの文字列が表示されているときは，その行にカーソル座標を設定することはできません．

〔例〕

```
10 CLS
20 FOR I=1 TO 10
30 LOCATE I,Y
40 PRINT  "123"
50 Y=Y+1
60 NEXT I
70 END
```

```
 123
  123
   123
    123
     123
      123
       123
        123
         123
          123
```

3.4.7 PSET（ピーセット：point set）

〔機　能〕

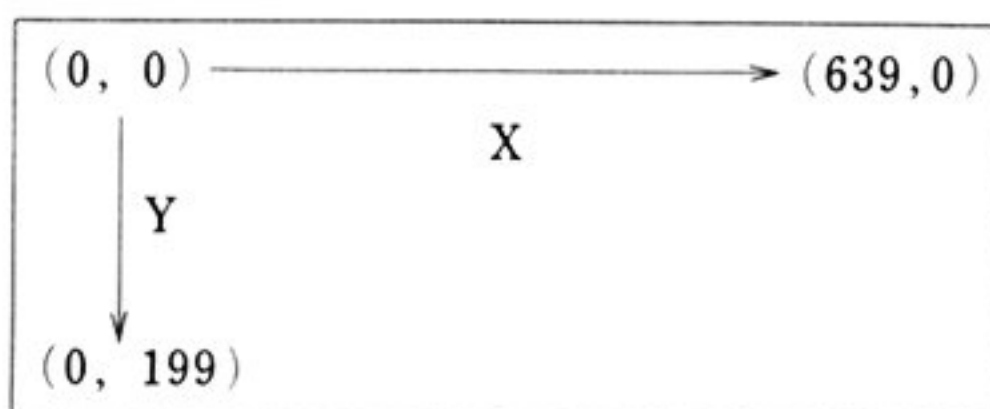
画面上の任意の位置にドットを設定します．

〔形　式〕

PSET（水平位置，垂直位置
　　[,[パレットコード][,機能]]）

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・水平位置，垂直位置は表示するドットのグラフィック座標を示します．ドットの座標は整数，実数のいずれでも指定できますが，実数型のときは，小数部を四捨五入した整数値がとられます．
　水平位置の範囲は，0から639までで，垂直位置は，0から199までです．
・パレットコードは，その座標に表示するドットの色を示し，パレットコードが省略されたときは，直前のCOLOR文のフォアグラウンドカラーで表示されます．
・機能としては，AND，OR，XORの指定ができます．このいずれかの機能が指定されたときは，現在画面に表示されている指定ドットのパレットコードとオペランドで指定されたパレットコードとの論理演算を行い，その結果のカラーで表示されます．
・画面上のドットの座標

〔例〕

```
5 COLOR 2,0
10 CLS
20 PSET(100,50,2,XOR)
30 PSET(540,50,3,XOR)
40 PSET(540,150,4,XOR)
50 PSET(100,150,5,XOR)
60 END
```

PSET実行後，(100，50)，(540，50)，(540，150)，(100，150) の位置にそれぞれ，赤，紫，緑，水色のドットがセットされます．

3.4.8 PRESET （プリセット：point reset）

〔機　能〕

画面上の任意の位置のドットを背景色にします．

〔形　式〕

PRESET （水平位置，垂直位置）

〔バージョン〕 V1.0　V2.0　V3.0

〔例〕

```
10 COLOR 7,0
20 CLS
30 PSET(320,100,2,OR)
40 FOR I=0 TO 200
50 NEXT I
60 PRESET(320,100)
70 FOR I=0 TO 200
80 NEXT I
90 GOTO 30
```

30行のPSET文と60行のPRESET文の組み合わせにより，（320，100）の位置で赤いドットが点滅します．

3.4.9 LINE（ライン：line）

〔形式1〕

LINE　[@][(x_1, y_1)]−(x_2, y_2),
　　文字列[,[パレットコード][,{B / BF}]]

〔機　能〕

画面にキャラクタを使って，線および箱を表示します．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・文字列の先頭文字を使ってキャラクタ座標（x_1, y_1）から（x_2, y_2）まで線を引きます．x_1，x_2は水平位置を示し，0から（1行の文字数－1）までの値を指定できます．y_1，y_2は垂直位置を示し，0から（画面の行数－1）まで指定できます．
・パレットコードは，表示する文字の色を示します．
・Bを指定すると，（x_1, y_1）と（x_2, y）を結ぶ線を対角線とする四角形の箱を表示します．BFを指定すると，箱の中も文字でうめます．
・パレットコードを省略したときは，直前のCOLOR文で指定したフォアグラウンドカラーで表示されます．
・（x_1, y_1）を省略したときは，直前に実行されたLINE文の（x_2, y_2）が（x_1, y_1）となります．
・文字列が空文字列のときは，ヌルコード（&H00）で書かれます．

〔例〕

```
10 CLS
20 X=20:Y=5
30 LINE@ (X-15,Y)-(X,Y),"H"
40 LINE@ -(X+10,Y+5),"B",1,B
50 LINE@ (31,8)-(51,20),"C",2,BF
60 LINE@ (0,8)-(28,18),"N",3
70 LOCATE 0,21
80 END
```

```
     HHHHHHHHHHHHHHHBBBBBBBBBBB
                    B         B
                    B         B
NNN                 B         BCCCCCCCCCCCCCCCCCCCCC
  NNNN              B         BCCCCCCCCCCCCCCCCCCCCC
     NNN            BBBBBBBBBBBCCCCCCCCCCCCCCCCCCCCC
       NNNN                    CCCCCCCCCCCCCCCCCCCCC
          NNNN                 CCCCCCCCCCCCCCCCCCCCC
             NNN               CCCCCCCCCCCCCCCCCCCCC
               NNNN            CCCCCCCCCCCCCCCCCCCCC
                  NNNN         CCCCCCCCCCCCCCCCCCCCC
                     NNN       CCCCCCCCCCCCCCCCCCCCC
                       NNNN    CCCCCCCCCCCCCCCCCCCCC
                          NNN  CCCCCCCCCCCCCCCCCCCCC
                               CCCCCCCCCCCCCCCCCCCCC
                               CCCCCCCCCCCCCCCCCCCCC

Ready
```

〔形式 2〕

```
LINE  [@][(x1, y1)]-(x2, y2),
      機能[,[パレットコード][, {B|BF}]]
```

〔機　能〕

画面にドットで線や箱を書きます．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・$(x_1,\ y_1)$ および $(x_2,\ y_2)$ は，画面上のドットのグラフィック座標を示し，2つの座標を結ぶ線を書いたり線を消したりします．

・機能は，PSET，PRESET，AND，OR，XORのいずれかを指定できます．
PSETを指定したときは，指定色で線を書き，PRESETを指定したときは，2つの座標を結ぶ線を消します．AND，OR，XORの指定をしたときは，指定色と現在画面に表示されているドットのパレットコードとの論理演算を行い，その結果のパレットコードで線を書きます．

・Bの指定をすると，2つの座標を結ぶ線を対角線とする四角形の箱を書き，BFの指定をすると，その箱の中を塗りつぶします．

・パレットコードを省略したときは，直前のCOLOR文のフォアグラウンドカラーで表示されます．

〔例〕

```
10 CLS
20 X=100:Y=50
30 LINE@ (X,Y)-(200,Y),PSET
40 LINE@ -(300,100),PSET,2,B
50 LINE@ (320,50)-(500,150),PSET,4,BF
60 LINE@ (380,20)-(520,130),XOR,2,BF
70 LINE@ (10,60)-(180,100),PSET,5
80 STOP
90 LINE@ (10,60)-(180,100),PRESET
100 LINE@ (380,20)-(520,130),XOR,4,BF
```

行番号50と60で重なった四角形を書きます．行番号80のSTOP文で実行が停止します．CONTコマンドを入力して，実行を再開すると（10，60）－（180，130）の線が消え，重なった四角形の色が変わります．

3.4.10 CONNECT (コネクト：connect)

〔機　能〕

指定座標間を直線で結びます．

〔形　式〕

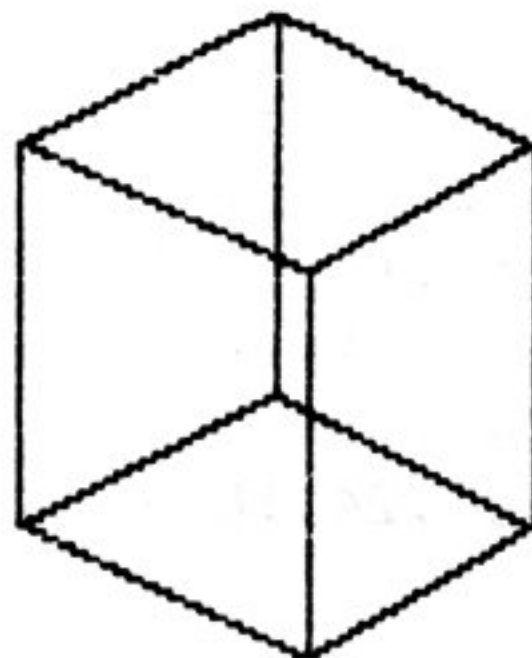
CONNECT　$(x_1, y_1)-(x_2, y_2)[-(x_3, y_3)]\cdots$
[,[パレットコード][,機能]]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・n 個の座標間を指定された順に次々と直線で結びます．
・パレットコードは，結ぶ直線の色を示し，指定のないときは，直前に実行された COLOR 文のフォアグラウンドカラーで表示されます．
・機能は，PSET，PRESET，AND，OR，XOR のいずれかです．

〔例〕

```
5 CLS
10 CONNECT (320,30)-(400,50)-(330,70)-
   (240,50)-(320,30)-(320,90)-(400,110)-
   (330,130)-(240,110)-(320,90),5,PSET
20 CONNECT (400,50)-(400,110),5,PSET
30 CONNECT (330,70)-(330,130),5,PSET
40 CONNECT (240,50)-(240,110),5,PSET
50 END
```

3.4.11 SYMBOL (シンボル：symbol)

〔機　能〕

画面上の任意の位置に文字列を指定角度，指定サイズで表示します．

〔形　式〕

SYMBOL　(x, y), 文字列, 横倍率, 縦倍率
[,[パレットコード][,[角度コード]
[,機能]]]

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・(x, y)は，文字列の表示を始める枠内の左上のドットのグラフィック座標を示します．
・横倍率および縦倍率は，表示する文字の横および縦の倍率を示し，倍率×8のドット数で画面に表示されます．
・角度コードは，文字を回転する角度を示し，省略時は0と見なされます．

0のとき，ノーマル
1のとき，90° 左回転
2のとき，180° 左回転
3のとき，270° 左回転

・パレットコードの指定がないときは，直前に実行されたCOLOR文のフォアグラウンドカラーで表示されます．
・機能として，PSET，PRESET，AND，OR，XOR，NOTの指定ができます．
・文字を表示するためのドット以外のドットは変化しません．

〔例〕

```
5 CLS
10 SYMBOL (316,93),"A",1,1
20 SYMBOL (323,108),"A",1,1,7,2
30 SYMBOL (331,97),"A",1,1,7,3
40 SYMBOL (308,104),"A",1,1,7,1
50 SYMBOL (268,80),"A",13,10,2
60 SYMBOL (224,50),"O",24,18,4
```

3.4.12 GET@（ゲット・アットマーク：get @）

〔形式 1〕

GET@　$(x_1, y_1)-(x_2, y_2)$，配列名

〔機　能〕

画面上のキャラクタを配列に読込みます．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・(x_1, y_1)，(x_2, y_2)はキャラクタ座標で，2つの座標を結ぶ線を対角線とする四角形内の文字を読込みます．

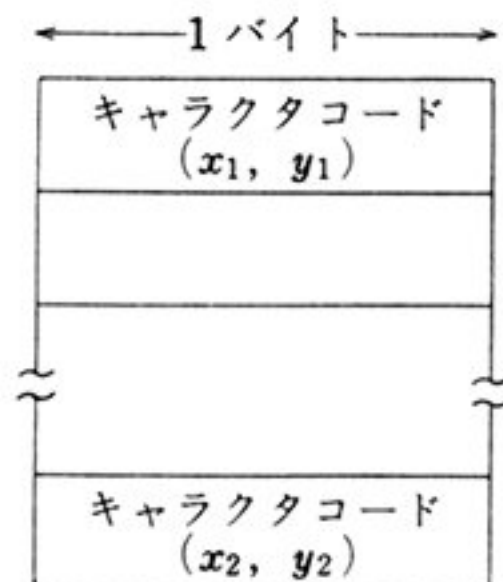
・配列の必要な大きさは，次のとおりです．

必要な配列の大きさ＝INT((キャラクタ単位の面積＋a－1)/a)

a は，配列の一要素の大きさです．

a：整数型　2
　単精度型　4
　倍精度型　8

なお，文字列の配列は“Illegal Fanction Call”のエラーになります．

・画面上のキャラクタコードが，配列に次のように格納されます．

配列内では，画面上の座標が左から右，上から下の順序で変化します．

〔例〕

```
10 CLS
20 LOCATE 10,5:PRINT "F-BASIC"
30 DIM A%(INT((2+2-1)/2))
40 DIM B%(INT((5+2-1)/2))
50 GET@(10,5)-(11,5),A%
60 GET@(12,5)-(16,5),B%
70 PUT@(15,8)-(15,9),A%
```

```
80 PUT@ (17,11)-(19,12),B%
90 LOCATE 0,21
100 END
```

表示された文字を配列に読取り，別の位置に表示(PUT @)します．

```
F-BASIC


          F
          -

           BAS
           IC
```

〔形式 2〕

GET @A $(x_1, y_1)-(x_2, y_2)$, 配列名

〔機　能〕

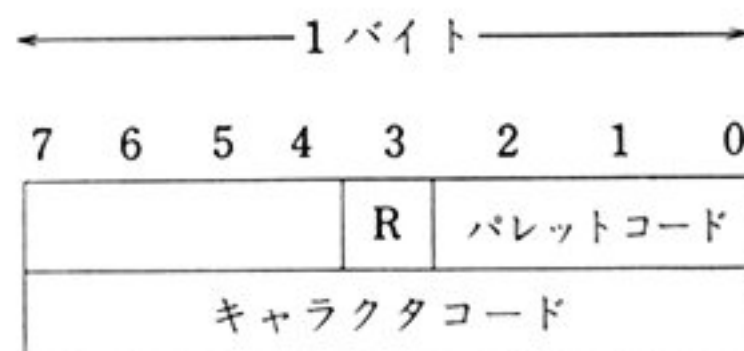
画面上のキャラクタとその属性を配列に読込みます.

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・GET @ A ステートメントは, キャラクタコードとそのパレットコードを読込みます.

・配列の大きさは, 形式1の場合の2倍必要です.

1つのキャラクタは2バイトの領域を占め, 上位バイトにそのキャラクタの属性, 下位バイトにキャラクタコードが格納されます. その形式は次のとおりです.

R　0：ノーマル
　　1：リバース

〔例〕

```
10 CLS
20 LOCATE 10,5
30 COLOR 2:PRINT "ABC";
40 COLOR 4:PRINT "DEF";
45 COLOR 1:PRINT "GHI"
50 DIM A%(INT((18+2-1)/2))
60 GET@A (10,5)-(18,5),A%
70 PUT@A (13,9)-(21,9),A%
80 COLOR 7,0
100 END

          ABCDEFGHI

Ready

             ABCDEFGHI
```

〔形式３〕

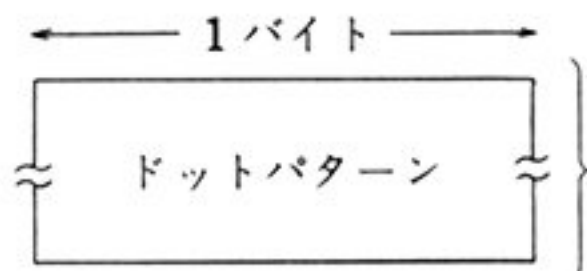
GET@　$(x_1,\ y_1)-(x_2,\ y_2)$，配列名，G
　　　[,パレットコード[,パレットコード]…]

〔機　能〕

画面上の指定色のドットパターンを配列に読込みます．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・$(x_1,\ y_1)$，$(x_2,\ y_2)$グラフィック座標で，２つの座標間を対角線とする四角形内のデータを読込みます．
・パレットコードは，最大８個の指定ができ，任意のドットの表示されているカラーが，指定されたパレットコードと一致しているならば，そのドットの対応する配列内のビットが１になります．
パレットコードの指定がないときは，任意のドットが背景色以外で表示されていれば，対応する配列内のビットが１になります．
・必要な配列の大きさは，次のとおりです．
　必要な配列の大きさ＝INT(INT((総ドット数＋7)/8)＋a－1)/a)
　　aは，配列の一要素の大きさです．
・次の形式で配列にデータが格納されます．

〔例〕

```
10 CLS
15 LINE@ (60,5)-(60,15),PSET,2
20 CONNECT (50,5)-(70,5)-(50,15)-(70,15)-(50,5),7,PSET
30 DIM A%(INT((INT((231+7)/8)+2-1)/2))
35 DIM B%(INT((INT((231+7)/8)+2-1)/2))
40 GET@ (50,5)-(70,15),A%,G,7
41 GET@ (50,5)-(70,15),B%,G,2
50 PUT@ (100,5)-(120,15),A%,PSET,7
60 PUT@ (80,20)-(100,30),A%,PSET,4
```

〔形式 4〕

GET@A　$(x_1, y_1)-(x_2, y_2)$, 配列名, G

〔機　能〕

画面上のグラフィックドットをそのカラー情報と共に配列に読込みます．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・(x_1, y_1), (x_2, y_2)はグラフィック座標で，二つの座標間を対角線とする四角形内のデータを読込みます．

・配列の大きさは，次のとおりです．

必要な配列の大きさ＝INT((INT((総ドット数＋7)/8)＊3＋a－1)/a)

a は，配列の一要素の大きさを示します．

・次の形式で配列にデータが格納されます．

〔例〕

```
10 CLS
15 COLOR 7,1
16 CLS
20 LINE(200,20)-(240,40),PSET,2,BF
30 LINE(200,20)-(220,29),XOR,1,BF
40 LINE(240,40)-(220,31),XOR,4,BF
50 LINE(220,20)-(240,30),XOR,6,BF
60 DIM A%(INT((INT((861+7)/8)*3+2-1)/2))
70 GET@A(200,20)-(240,40),A%,G
80 PUT@A(100,70)-(140,90),A%,NOT
90 PUT@A(150,70)-(190,90),A%,PSET
100 PUT@A(200,70)-(240,90),A%,OR
110 PUT@A(250,70)-(290,90),A%,XOR
120 PUT@A(300,70)-(340,90),A%,AND
130 LOCATE 0,20
140 END
```

3.4.13 PUT@ (プット・アットマーク：put @)

〔形式1〕

PUT@ $(x_1, y_1)-(x_2, y_2)$,
配列名[,パレットコード]

〔バージョン〕 V1.0 V2.0 V3.0

〔機 能〕

GET@文（形式1）で読込まれたキャラクタを画面の任意の位置に表示します．

〔説 明〕

・配列の形式および内容は，GET @文の形式1で述べた形式になっていなければなりません．

・パレットコードは，表示する文字の色を示し，この指定のないときは，直前のCOLOR文のフォアグラウンドカラーで表示されます．

〔例〕

3.4.12 GET @ 文の形式1を参照して下さい．

〔形式2〕

PUT@A　$(x_1, y_1)-(x_2, y_2)$,配列名

〔機　能〕

GET@A文（形式2）で読込まれたキャラクタを画面上の任意の位置に表示します.

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・配列の形式および内容は, GET @ A文の形式2で述べた形式になっていなければなりません.

〔例〕

3.4.12　GET @ 文の形式2を参照して下さい.

〔形式3〕

PUT@　$(x_1, y_1)-(x_2, y_2)$,配列名，機能
　　　[,パレットコード]

〔機　能〕

GET@文（形式3）で読込まれたグラフィックドットを任意の画面に表示します．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・配列の形式および内容は，GET@文の形式3で述べた形式になっていなければなりません．

・配列の中には表示されるドットのカラー情報は持ってなく，そのドットが表示されるか否かの情報を持っているだけです．従って，表示するドットの色をパレットコードで指定します．このパレットコードを省略したときは，直前に実行されたCOLOR文のフォアグラウンドカラーが採用されます．

・機能として，PSET，PRESET，AND，OR，XOR，NOTのいずれかの指定ができます．

PSETを指定しますと，配列データのビットがONのドットを指定色で表示し,OFFのドットは変化しません．

PRESETを指定しますと，配列データのビットがONのドットを背景色にします．

NOTの指定をしますと，ビットがOFFのドットを指定色で表示します．

AND，OR，XORの指定をしますと，配列データのビットがONのドットに対し，指定パレットコードと画面上のドットのパレットコードの論理演算を行い，その結果のパレットコードで表示します．ビットがOFFのドットは変化しません．

〔例〕

3.4.12　GET@文の形式3を参照して下さい．

〔形式4〕

PUT@A　$(x_1, y_1)-(x_2, y_2)$,配列名，機能

〔機　能〕

GET@A文（形式4）で読込まれたグラフィックデータを任意の画面に表示します．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・配列の形式および内容は,GET@A文の形式4で述べた形式に従っていなければなりません．

・機能には，AND，OR，XOR，NOT，PSETのいずれかが指定できます．
AND，OR，XORを指定したときは，現在画面に表示されているドットのB・R・Gのドットパターンと配列で示されたB・R・Gのドットパターンとの論理演算を行い，その結果のドットパターンが画面に表示されます．
NOTを指定したときは，配列のB・R・Gの全ビットを反転させ，その結果を画面に表示します．従って，配列に読込まれている色の補色で表示されることになります．
PSETを指定したときは，配列データをそのまま画面に表示します．

〔例〕

3.4.12　GET@A文の形式4を参照して下さい．

3.4.14 CIRCLE（サークル：circle）

〔機　能〕

画面上の任意の座標を中心点として，円または円弧を描きます．

〔形　式〕

```
CIRCLE[@]  (x, y),
           半径[,[パレットコード][,[比率]
           [,[開始位置][,[終了位置]
           [ , [{F/N}][,機能]]]]]
```

〔バージョン〕　V1.0　V2.0　V3.0

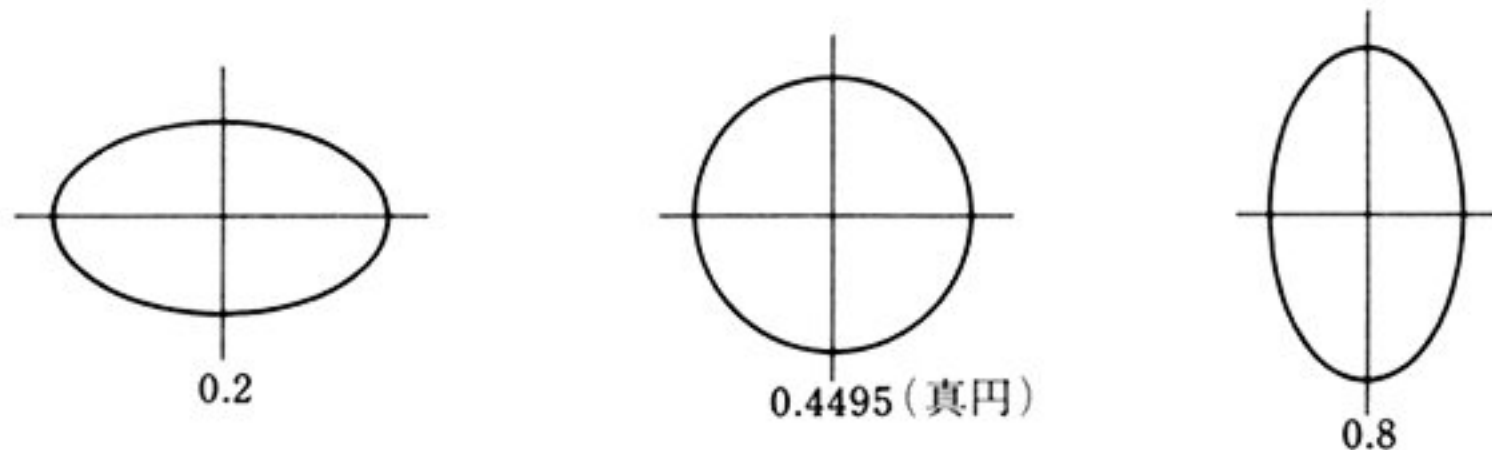
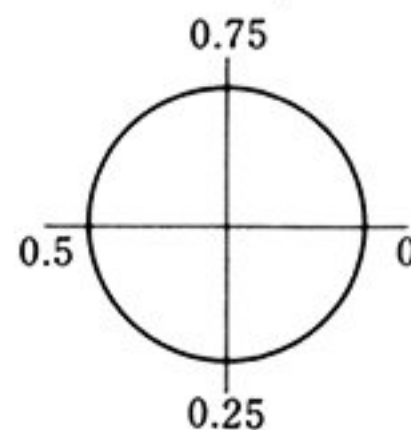
〔説　明〕

・(*x*, *y*) は，中心点のグラフィック座標を示します．

・半径は，X軸上での半径でありドットの個数で示します．

・パレットコードは，描く円または円弧の色です．省略したときは，直前のCOLOR文のフォアグラウンドカラーが用いられます．

・比率は，X軸上のドット数に対するY軸上のドット数の比率を示し，0～1の範囲で指定します．なお，指定のないときは0.4495がとられます．

0.4495（真円）

・開始位置は，描き始める位置を示し，0～1の範囲で指定します．

・終了位置は，描き終わる位置を示し，0～1の範囲で指定します．開始位置＝終了位置のとき，完全な円が描かれます．なお，開始位置および終了位置の省略は0と見なされます．

・Fの指定をすると，円の内部を塗りつぶし，Nの指定をすると，塗りつぶしません．省略したときは，Nと見なされます．

・機能としては，PSET，PRESET，AND，OR，XORの指定ができます．
この指定が省略されたときには，PSETが指定されたものと見なされます．

・開始位置及び終了位置の値として認識される最小値は0.0156です．したがって，円周の64等分の1まで区分することができますが，それ以下の円孤を描くことはできません．

・Fの指定をしたときは，中心点から円周上の点へ線を引くことにより円の内部が塗りつぶされますが，PSET，PRESET，AND，OR，及びXORは，その線を引くときに効果を持ちます．したがって，XORを用いるとしま模様の円が描かれます．

〔例〕

```
10 CLS
20 CIRCLE(60,90),20,2,.9,,,F
30 CIRCLE(150,90),50,4,.2,,,N
40 CIRCLE(280,90),40,5,.4495,0,.3,F
50 CIRCLE(400,90),50,7,.4495,0,0,N
60 CIRCLE(530,90),50,3,.6,0,.9,N,PSET
70 END
```

注）画面とプリンタでは，X方向，Y方向のドット間隔が異ります．

3.4.15 GCURSOR（ジー・カーソル：graphic cursor）

〔機　能〕

画面上のグラフィックカーソルのドット座標を読取ります．

〔形　式〕

```
GCURSOR  (x,y),(変数名1, 変数名2)
         [,(変数名3,変数名4)……]
         [,パレットコード]
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・(x, y) は，最初にグラフィックカーソルを表示する座標です．グラフィックカーソルは，指定位置に指定したパレットコードで＋印で示されます．

・(変数名，変数名) は，現在表示されているグラフィックカーソルのドット座標を読込む領域です．座標の読込みは，リターンキーが押されたとき行われます．

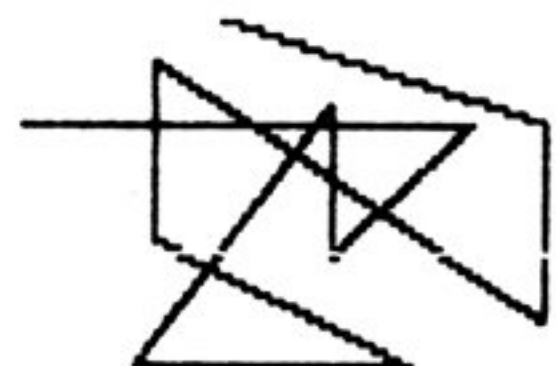
・グラフィックカーソルの移動は，カーソル移動キーで行い，指定した n 個のドット座標が読終わると，表示されていた＋印が消えます．
移動ドット数は次のようになります．

  ◦ カーソル移動キーを押したとき，その方向に1ドット移動します．
  ◦ 数字キー(0 ～ 9)を押すと，その後のカーソル移動キーによる移動ドット数は，その数字キーの数になります．
  ◦ SHIFT キーを併用すると，20ドット単位で移動します．

・パレットコードを省略したときは，直前に実行されたCOLOR文のフォアグラウンドカラーで表示されます．

・読取れる座標の個数は，10個までです．

〔例〕

```
10 CLS
20 GCURSOR (320,100),(X,Y)
30 LINE@(0,0)-(X,Y),PRESET
35 FOR I=1 TO 10
40 GCURSOR (X,Y),(X,Y)
50 LINE@-(X,Y),PSET
60 NEXT I
70 END
```

3.4.16 PAINT (ペイント：paint)

〔機　能〕

指定された境界色で囲まれた範囲を塗りつぶします.

〔形　式〕

PAINT (水平位置,垂直位置)[,[パレットコード]
[,境界色1[,境界色2]…]]

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・(水平位置, 垂直位置) で塗り始めるドットのグラフィック座標を指定します.
・パレットコードは塗りつぶす色を示し, 省略したときは, 直前の COLOR 文のフォアグラウンドカラーで塗られます.
・境界色により, 囲む境界の色を指定し, 複数個の境界色を指定することができます.
境界色の指定は, パレットコードで行います.
なお, パレットコード自身も境界色となります.
・境界色の指定がないときは, 指定されたパレットコードが境界色となります.
・塗りつぶす範囲が細かく分割されているときは, 作業領域の大きさの制限により, 一部塗り残しが出る場合があります.

〔例〕

```
10 CLS
12 C=0
15 Y=0
20 FOR X=0 TO 200 STEP 25
30 LINE(320-X,100-Y)-(320+X,100+Y),PSET,C,B
40 Y=Y+10
50 C=C+1
60 NEXT X
70 PAINT(320,109),2,1
80 PAINT(380,129),6,2,3
90 PAINT(430,149),1,4,5
100 PAINT(480,169),4,6,7
110 PAINT(455,159),3,5,6
120 PAINT(405,139),7,3,4
130 PAINT(355,119),5,1,2
140 END
```

行番号20から60のFOR～NEXTループの中で，7つの箱を書き，行番号70から130のPAINT文でその中を塗りつぶします．

3.5　音楽演奏機能

3.5.1　PLAY　（プレイ：play）

〔機　能〕

音楽の演奏を行います．

〔形　式〕

```
PLAY　"MML"［，"MML"［，"MML"］］
```

〔バージョン〕　V1.0　V2.0　**V3.0**

〔説　明〕

MML（Music Macro Language）を用いて自動演奏を行います．

MMLはテンポ，音長，音程などを指定する一種の言語です．

そしてMMLを"，"で区切ることにより3和音まで演奏を行うことができます．

MMLには以下のコマンドがあります．

(1)　音程（A，B，C，D，E，F，G）

このA～Gまでのコマンドは音程を表わします．これはそれぞれA＝ラ，B＝シ，C＝ド，D＝レ，E＝ミ，F＝ファ，G＝ソに対応しています．

また，このコマンドの後に♯，＋(♯)，－(♭)をつけ加えて半音を表わすことができます．

(2)　休符（R）

このコマンドは休符を指定します．

(3)　音長（Ln）

このコマンドは音長を指定します．nは1～64の整数で1の時を最大とした音分数の逆数が値となります．このコマンドから以降は，指定した音長で演奏されます．また音程，休符のコマンドの後にnを追加すると，その音だけその音長で演奏されます．

例　4分音符のド………L4C　　または　C4

16分音符のファの♯……L16F♯　または　F♯16　または　F＋16

全音符のソの♭………L1G－　または　G－1

音長の決め方の簡単な方法は，n分音符ならそのnが求める値です．

しかし，3連符の時は以下のように求めます．

その他，ふ点は，A～Gコマンドの音長の後にピリオドを追加することで表わします．ふ点はその音長の$\frac{3}{2}$倍の長さになるので〝A 4‥〟にすると音長4（4分音符）の$\frac{9}{4}$倍で音長9の長さで演奏されます．

音長対応表

音程			休符	
記号	名称	n	記号	名称
𝅝	全音符	1	𝄻	全休符
𝅗𝅥	2分音符	2	𝄼	半休符
♩	4分音符	4	𝄽	4分休符
♪	8分音符	8	𝄾	8分休符
𝅘𝅥𝅯	16分音符	16	𝄿	16分休符

(4) オクターブ（On）

このコマンドはオクターブを指定します．nは1～8までの整数です．なお，基準周波数440Hzのラの音(A)はオクターブ4(O 4)です．

(5) 特殊音程（Nn）

音程はA～Gまでのコマンドで表わしますが，Nで指定することもできます．nは1～96までの整数です．これは，コマンドでO 1 Cを1とし，音程を12音程としてO 8 Bの96まで指定できます．

対応表

	O 1		O 2		O 3		O 4		O 5		O 6		O 7		O 8	
C	1	2	13	14	25	26	37	38	49	50	61	62	73	74	85	86
D	3	4	15	16	27	28	39	40	51	52	63	64	75	76	87	88
E	5		17		29		41		53		65		77		89	
F	6	7	18	19	30	31	42	43	54	55	66	67	78	79	90	91
G	8	9	20	21	32	33	44	45	56	57	68	69	80	81	92	93
A	10	11	22	23	34	35	46	47	58	59	70	71	82	83	94	95
B	12		24		36		48		60		72		84		96	

この表は，1オクターブごとにそれぞれ2列あります．左が普通の音，右がその音の半音上の音となります．

例1　O 4 A＃の時のn

Aコマンド列でO 4の所は46か47です．その半音上がっているので47が求める値です．

例2　O 3 B－の時のn

B- はシの半音下がるので，ラの半音上がりと同じ音程，つまりO 3 A＃を探せばいいので求める値は35です．

(6) テンポ（Tn）

このコマンドは，テンポを指定します．nは32～255の整数です．nは1分間に4分音符をn回数える早さです．

(7) 音量（Vn）

このコマンドは，そのパートの音量を指定します．nは0～15の整数で15で最大

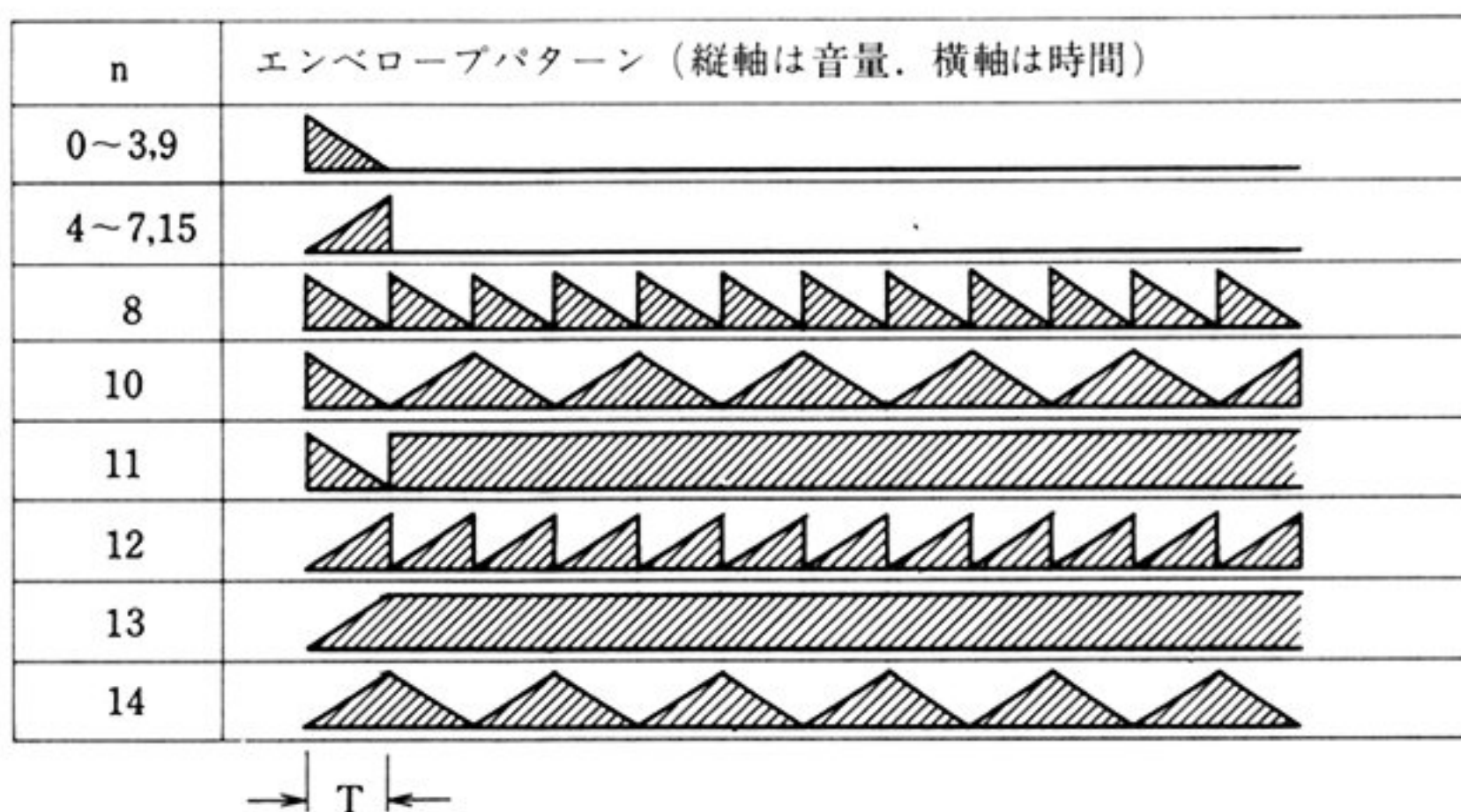
となります．また0ではそのパートの音は，出力されません．

(8) エンベロープパターン（Sn）

このコマンドは，エンベロープを指定します．nは0～15までの整数です．この指定により音量はエンベロープパターンに依存されるためVコマンドは無効となります．この指定の解除は再びVコマンドで音量を指定することにより行われます．

(9) エンベロープ周波数（Mn）

このコマンドはSコマンドとペアで使用します．nは0～65535までの整数で，エンベロープ周波数を指定します．エンベロープパターンと周波数の計算式を下に示します．

エンベロープパターンとSコマンドの数値nの対応表

周期Tとコマンド M の数値の関係式

$$T=\frac{256\cdot D}{f}\quad (f=1.2288\mathrm{MHz})$$

(10) 間接指定（＝変数名；）

今までのnは定数ですが，間接指定により，変数を使うことができます．

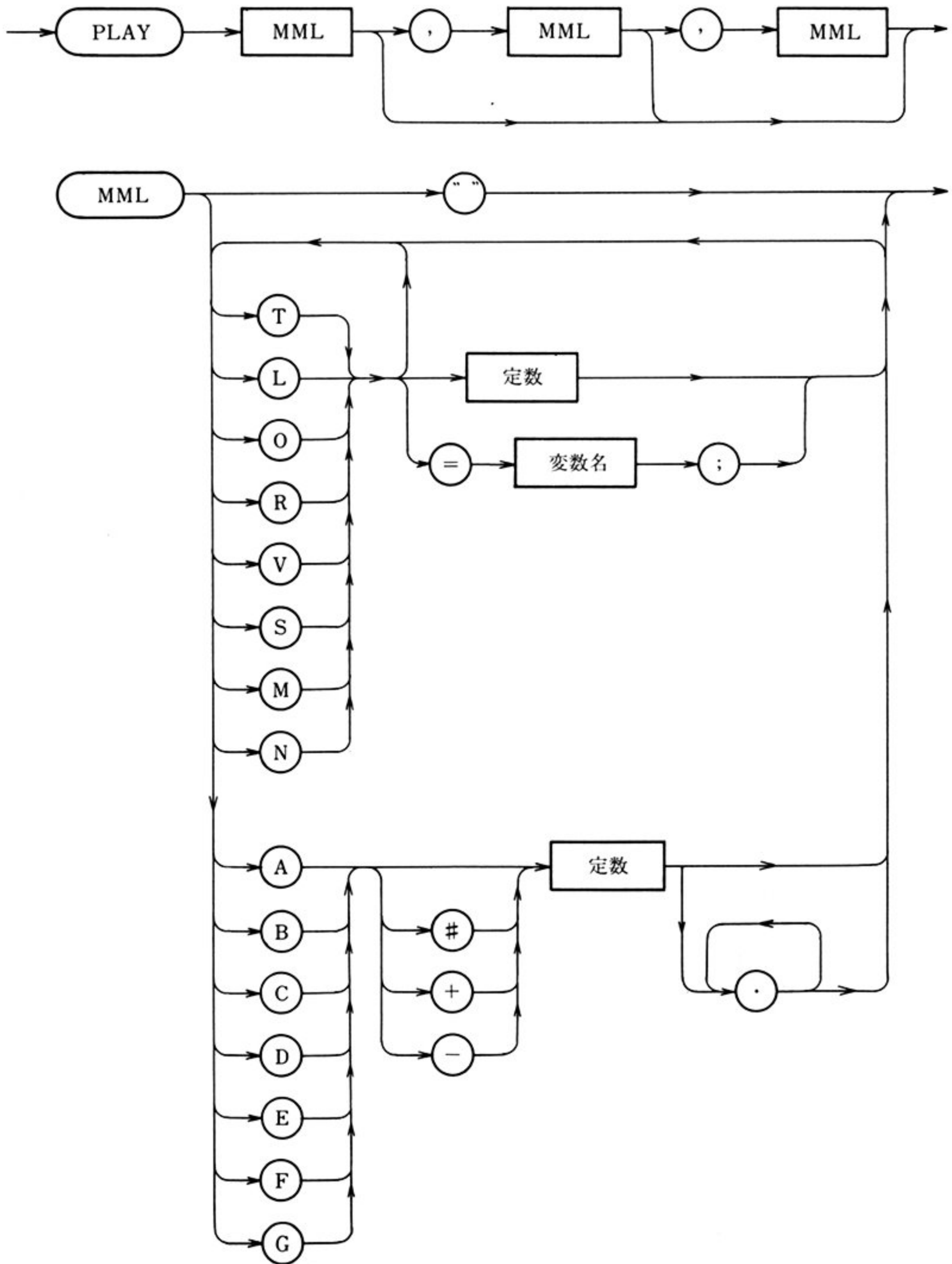
例　I＝10：PLAY〝O2L＝I;〞は

PLAY〝O2L10〞と同じ意味です．

(11) デフォルト

今までのnの中で省略（デフォルト）できるものがあります．下図にそれを示します．

コマンド	記号	デフォルト値	注意事項
音長	L	4	A～G，Rでの音長はLコマンドで指定した値による．
オクターブ	O	4	
テンポ	T	120	
音量	V	8	

PLAY構文図
PLAY
MML
,
MML
,
MML
MML
" "
T
L
O
R
V
S
M
N
定数
=
変数名
;
A
B
C
D
E
F
G
#
+
−
定数
.

〔注　意〕

(1) PLAY コマンドはオーディオカセット関係のコマンドと共用できません．もし〝CAS0:″をOPEN中にPLAYコマンドを実行した場合にはDevice In USE エラーが発生します．

(2) PLAY コマンドが中断する場合は次の通りです．
- ・BREAKキーを押した場合
- ・入力待ちでCTRL－X，CTRL－C を押した場合
- ・オーディオカセット関係のコマンド（MOTOR を除く）を使用した場合
- ・RUN コマンドを実行した場合
- ・Abort 表示された場合

(3) PLAY コマンドでは最高3声で16音までメモリーに記憶し割込処理によって音階を発生する為うまくプログラムすることにより，他の処理の実行しつつバックグランドで音楽の演奏を行えます．

(4) PLAY コマンドでバックグランド演奏中にSOUND コマンドを実行できますが，当然SOUND コマンドによるPSGのレジスタの書替えにより音が聞えなくなったりノイズがまじったりします．この場合 (2) の条件でPSGは初期状態に戻ります．

3.5.2 SOUND （サウンド：sound）

〔機　能〕

PSGを直接コントロールします．

〔形　式〕

SOUND　PSGのレジスタ番号，データ

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

SOUNDはPSG（Programmable Sound Generater）のレジスタの内容を直接書き変えてPSGを直接コントロールする命令です．PSGのレジスタ番号は0から13までの値でデータは0から255までの数です．各レジスタの働きは図1，図2の通りです．FM-7の内蔵しいるPSGはAY－3－8913タイプです．ブロック図を図1に示します．この図からPSGは3つのオシレータとノイズ・ジェネレータ，エンベロープ・ジェネレータ，ミキサー，そして3つのアッテネータを内臓していて，簡単なシンセサイザとしての機能を一通り備えていることがわかります．

図2にこの14個のレジスタの一覧表を示します．R_0からR_5は3つのオシレータの発振周波数を決定します．R_0とR_1，R_2とR_3，R_4とR_5をそれぞれ組みにして使用し，そのうちの下位12ビットの値が有効です．実際の発振周波数は次のように計算できます．

$$f=\frac{f_{clock}}{16\times D} \qquad D=\frac{f_{clock}}{16\times f}$$

ここでDは，書込むべきデータ，f_{clock}=1.2288MHz

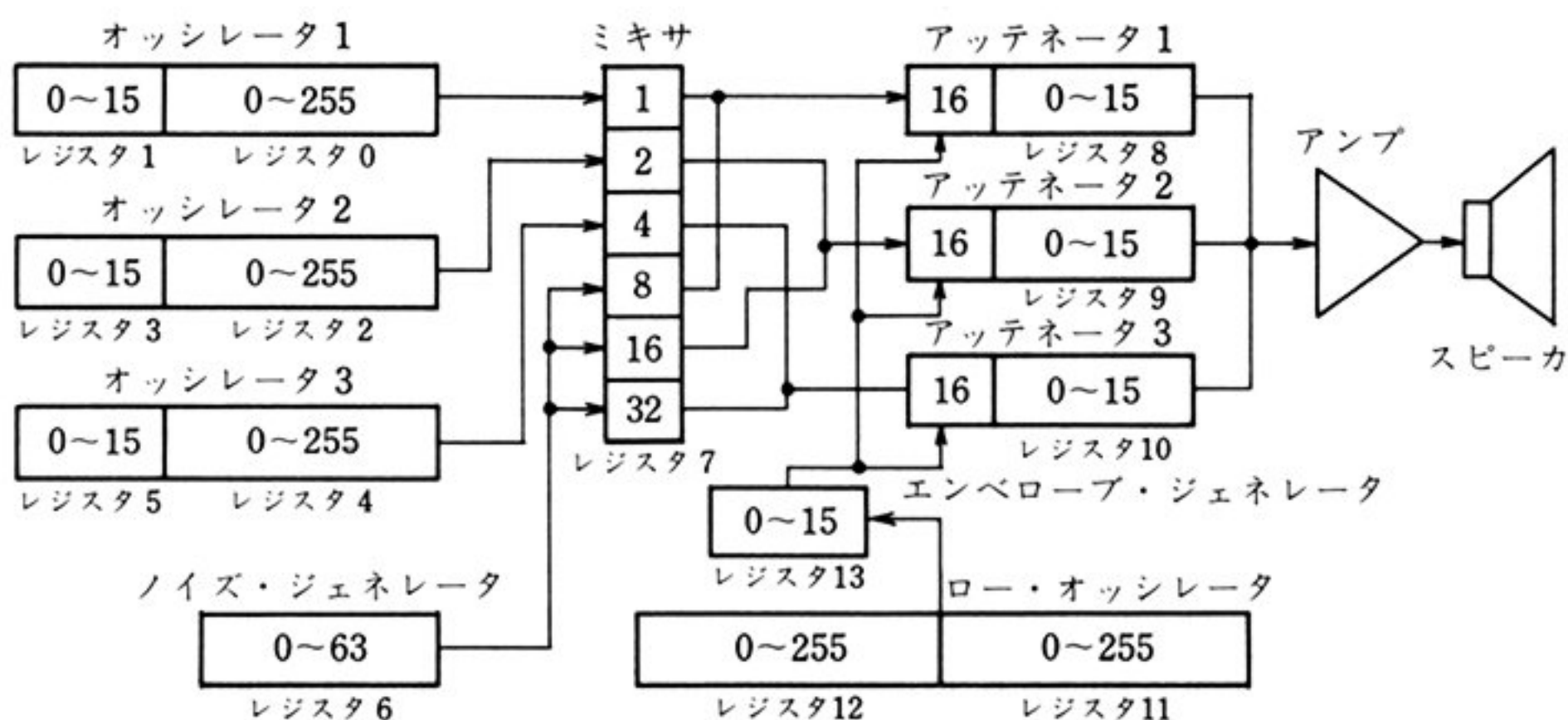



図1　PSGのブロック

レジスタ＼ビット	働き	7	6	5	4	3	2	1	0	
R_0	チャンネルAの音階	下位8ビット・データ								音階は12ビットのデータで表現する
R_1						上位4ビット・データ				
R_2	チャンネルBの音階	下位8ビット・データ								
R_3						上位4ビット・データ				
R_4	チャンネルCの音階	下位8ビット・データ								
R_5						上位4ビット・データ				
R_6	ノイズ周波数	5ビット・データ								ノイズの平均周波数を指定する
R_7	入出力の選択			ノイズ C	ノイズ B	ノイズ A	トーン C	トーン B	トーン A	各チャンネルから出す音源や，I/Oポートを指定する
R_8	チャンネルAの音量					4ビット・データ				M＝0のとき下位4ビットが音量調節 M＝1のときエンベロープ作動
R_9	チャンネルBの音量					4ビット・データ				
R_{10}	チャンネルCの音量					4ビット・データ				
R_{11}	エンベロープ周期	下位8ビット・データ								
R_{12}		上位8ビット・データ								
R_{13}	エンベロープ波形					CONT.	ATT.	ALT.	HOLD	

図2　PSGのレジスタ

楽器のチューニングなどに使われるラの音（440Hz）を出力するときは，D＝349をレジスタに書込みます．

R_6はノイズの平均周波数を決定するレジスタで，下位5bitのみが有効です．周波数は次の式で求められます．

$$f=\frac{f_{clock}}{16\times D}\qquad D=\frac{f_{clock}}{16\times f}$$

R_7はPSGのモード設定を行うレジスタです．対応する各bitを0にすることによりモードを設定できます．

R_8からR_{10}は音量の設定をするためのものです．下位4bitが有効で，最大音量にセットするには15を書込みます．ただし，このレジスタのbit 4を1にするとエンベロープがイネーブルになり，音量の指定は無視されます．この場合，音量のコントロールはエンベロープ・ジェネレータに任されることになります．

R_{11}からR_{13}はこのエンベロープのコントロールを行うジェネレータです．エンベロープの形式は図3にある中から選択できます．選択した波形コードをR_{13}に書込めば，その波形がセットされ，同時にそれがエンベロープのトリガとなります．エンベロープの長さは，エンベロープ周波数としてR_{11}とR_{12}で指定します．エンベロープ周波数は次式で与えられます．

$$f=\frac{f_{clock}}{256\times D}$$

図3の下にあるεの大きさはこのfの値から

$$\varepsilon=\frac{1}{f}$$

で求められます．

レジスタ R_{13} とエンベロープ・パターン

レジスタR_{13}の下位4ビット				エンベロープ・パターン（縦軸は音量，横軸は時間）
0	0	−	−	
0	1	−	−	
1	0	0	0	
1	0	0	1	
1	0	1	0	
1	0	1	1	
1	1	0	0	
1	1	0	1	
1	1	1	0	
1	1	1	1	

ε

3.6 プログラマブル・ファンクションキー機能

3.6.1 KEY (キー:key)

〔機 能〕

プログラマブル・ファンクションキーに文字列を定義します.

〔形 式〕

KEY ファンクションキー番号, 文字列

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・ファンクションキー番号は, キーボード上のPFキーの番号に対応し, 1〜10の値でなければなりません.

・文字列は, 最大15文字までの文字またはコントロール文字で構成し, コントロール文字は, CHR$関数を加算記号で継いだ文字式で指定します.

〔例〕

```
KEY 1,"LOCATE"
```

ファンクションキー1にLOCATEを定義します.

3.6.2 KEY LIST (キー・リスト：key list)

〔機 能〕

プログラマブル・ファンクションキーに定義される文字列を画面に表示します.

〔形 式〕

```
KEY LIST
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・電源投入時およびシステムリセット時には，次のようにプログラマブル・ファンクションキーの文字列が設定されています.

(1) V1.0, V2.0の場合

PF 1	AUTO␣
PF 2	LIST (CR)
PF 3	RUN (CR)
PF 4	CONT (CR)
PF 5	LLIST (CR)
PF 6	LOAD (CR)
PF 7	SAVE″
PF 8	?DATE $, TIME $ (CR)
PF 9	SCREEN 7, 7 (CR)
PF 10	HARDC (CR)

(2) V3.0の場合

	ROMモード	Diskモード
PF 1	AUTO␣	AUTO␣
PF 2	LIST (CR)	LIST (CR)
PF 3	RUN (CR)	RUN (CR)
PF 4	CONT (CR)	CONT (CR)
PF 5	LLIST (CR)	LLIST (CR)
PF 6	LOAD (CR)	LOAD″
PF 7	SAVE″	SAVE″
PF 8	?DATE $, TIME $ (CR)	FILES
PF 9	SCREEN 7,7 (CR)	SCREEN 7,7 (CR)
PF 10	HARDC (CR)	HARDC (CR)

3.6.3 KEY(n) ON/OFF/STOP (キー・オン/オフ/ストップ：key(n) on/off/stop)

〔機 能〕

ファンクションキーからの割込みの許可，禁止，停止をします．

〔形 式〕

KEY(ファンクションキー番号) {ON / OFF / STOP}

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・KEY（n）ONにより，ファンクションキー*n*からの割込み動作を可能にします．この文を実行後，ファンクションキーnを押すと，ON KEY（n）GOSUB文により定義されている割込み処理ルーチンが呼出されます．
・KEY(n) OFFにより，ファンクションキーからの割込みを禁止します．
・KEY(n) STOPは，割込み動作を一時停止します．この文を実行後にファンクションキーを押すと，その時点では割込み処理ルーチンの呼出しは行われず，次にKEY(n) ONを実行することにより割込みルーチンが呼出されます．
・RUNコマンドによりすべてのファンクションキーがOFF状態になります．
・KEY ONの状態では，ファンクションキーに定義されている文字列の入力はできませんので，割込み処理終了後，KEY OFFを実行する必要があります．

〔例〕

```
1 ON KEY (1) GOSUB 60
5 KEY (1) ON
10 CLS
20 C=1
30 COLOR C
40 PRINT "■";
50 GOTO 40
60 BEEP
70 C=C+1
80 IF C=8 THEN C=1
90 RETURN 30
```

行番号40と50でつねに同じ色で，"■"を書いていますが，ファンクションキー1が押されると，カラーコードを1加算して，色を変えています．

3.6.4 ON KEY(n) GOSUB (オン・キー・ゴーサブ：on key(n) go to subroutine)

〔機　能〕

ファンクションキー割込みルーチンの定義をします．

〔形　式〕

ON　KEY(ファンクションキー番号)　GOSUB
　　行番号

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・行番号は，割込み処理ルーチンの開始行番号です．
・PF－nのキーが押されたとき，KEY（n）がONの状態なら，実行を中断してON KEY(n) GOSUBにより,指定された行番号に制御が渡ります．
・割込みルーチンからの復帰は，RETURN文により行われ，そのRETURN文に行番号があるときは，その行から実行が再開されます．
行番号がないときは，中断した個所から再開されます．

〔例〕

3.6.3　KEY（n）　ON／OFF／STOP文を参照して下さい．

3.7 タイマ割込み機能

3.7.1 ON TIME GOSUB (オン・タイム・ゴーサブ：on time go to subroutine)

〔機 能〕

タイマ割込みルーチンの定義をします．

〔形 式〕

ON TIME GOSUB 行番号

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・行番号は割込み処理ルーチンの開始行番号です．
・割込みルーチンからの復帰は，RETURN 文により行われます．

〔例〕

```
1 WIDTH 40,25
5 CLS
10 TIME$="07:29:50"
20 ON TIME GOSUB 200
30 TIME "07:30:00"
40 TIME ON
50 LOCATE 15,10:PRINT TIME$:GOTO 40
60 END
200 BEEP 1
205 LOCATE 15,10:COLOR 2:PRINTTIME$
210 COLOR 7:LOCATE 10,12:INPUT "ｵﾒｻﾞﾒﾉ ｼﾞｶﾝﾃﾞｽ !!!",A$
220 BEEP 0
230 RETURN 60
```

このプログラムでは，TIME$ によりタイマの時刻を7時29分50秒に設定し，7時30分になったとき割込みを行うよう設定しています．所定の時刻になるまでは，タイマの時刻を表示していますが，所定の時刻になると割込み処理ルーチンでメッセージを出力し，ブザーを鳴らします．ここでリターンキーを押すとブザーを止めた後ストップします．

3.7.2 TIME (タイム：time)

〔機　能〕

タイマ割込みの時刻を設定します．

〔形　式〕

TIME　"hh：mm：ss"

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・hh：mm：ss により割込み時刻を時分秒で指定します．
・タイマ割込みの設定は，1レベルのみ行うことができ，同時に2つの時刻の設定はできません．

〔例〕

3.7.1 ON TIME GOSUB 文を参照して下さい．

3.7.3 TIME ON/OFF/STOP (タイム・オン/オフ/ストップ：time on/off/stop)

〔機　能〕

タイマ割込みの許可，禁止，停止をします．

〔形　式〕

```
TIME {ON | OFF | STOP}
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

- TIME ON により，TIME 文で指定された時刻での割込み動作を可能にします．この文を実行後，TIME 文で指定された時刻になると，プログラムの実行を中断して，ON TIME GOSUB 文で定義されている割込み処理ルーチンに制御が移されます．割込みルーチンからの復帰は RETURN 文により行われ，RETURN 文に行番号があるときは，その行から実行が再開されます．もし行番号がないときは，プログラムの中断した個所から実行が再開されます．
- TIME OFF により，タイマ割込みを禁止します．
- TIME STOP は，割込み動作を一時停止します．この文を実行後は，指定時刻での割込みは受付けられますが，割込み動作はその時点で行われません．次に TIME ON 文が実行されたときに割込みルーチンの呼出しが行われます．

〔例〕

3.7.1 ON TIME GOSUB 文を参照して下さい．

3.7.4 ON INTERVAL GOSUB（オン・インターバル・ゴーサブ：on interval go to subroutine）

〔機　能〕

インターバルタイマ割込みルーチンの定義をします.

〔形　式〕

ON INTERVAL GOSUB 行番号

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・行番号は割込み処理ルーチンの開始行番号です.

・割込みルーチンからの復帰は，RETURN 文により行われます.

〔例〕

```
10 ON INTERVAL GOSUB 50
15 INTERVAL 2
20 TIME$="10:00:00"
30 INTERVAL ON
40 GOTO 40
50 BEEP
60 I=I+1
70 PRINT TIME$,I
80 RETURN
```

このプログラムでは，インターバルとして，2を設定しています.
従って，2秒を経過すると割込み処理ルーチンへ分岐し，タイマの内容をプリントして，元のプログラムへ戻ります.

```
RUN
10:00:02        1
10:00:04        2
10:00:06        3
10:00:08        4
10:00:10        5
10:00:12        6
10:00:14        7
10:00:16        8
10:00:18        9
10:00:20        10
```

3.7.5 INTERVAL (インターバル：interval)

〔機 能〕

インターバルタイマ割込みの間隔を指定します.

〔形 式〕

```
INTERVAL 割込み間隔
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・割込み間隔の単位は秒であり，最大65535秒までの指定ができます.

・INTERVAL 文を実行すると，指定された秒間隔で繰返し割込みが発生し，割込み許可状態であれば，そのつど ON INTERVAL GOSUB 文により定義されている割込み処理ルーチンへ制御が渡されます.

・経過時間のカウントは，INTERVAL 文を実行した直後，および割込み発生直後から行われ，指定した秒数に達したとき，次の割込みが発生します.

・指定された割込み間隔が整数でないときは，小数点以下は切捨てられ整数に変換されます.

〔例〕

3.7.4 ON INTERVAL GOSUB 文を参照して下さい.

3.7.6 INTERVAL ON/OFF/STOP (インターバル・オン/オフ/ストップ：interval on/off/stop)

〔機　能〕

インターバルタイマ割込みの許可，禁止，停止をします．

〔形　式〕

INTERVAL {ON | OFF | STOP}

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・INTERVAL ON により，INTERVAL 文で指定された秒間隔での割込み動作を可能にします．

・INTERVAL OFF により，インターバルタイマ割込みを禁止します．

・INTERVAL STOP により，割込み動作を一時停止します．この文の実行後は，割込みは受付けられますが，割込み動作は行われません．次に INTERVAL ON 文が実行されたときに割込みルーチンの呼出しが行われます．

・プログラム終了時には，INTERVAL OFF を実行する必要があります．

〔例〕

3.7.4 ON INTERVAL GOSUB 文を参照して下さい．

3.8 回線制御機能

3.8.1 ON COM(n) GOSUB (オン・コム・ゴーサブ：on comnunication(n) go to subroutine)

〔機 能〕

通信回線からの入力割込み時の処理ルーチンを定義します．

〔形 式〕

ON COM(ポート番号) GOSUB 行番号

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・行番号は，ポート番号で指定した通信ポートから入力があったときに呼出される割込み処理ルーチンの開始行番号です．

・ポート番号は0～4の値でなければなりません．

・割込みルーチンからの復帰はRETURN文により行われます．

・回線制御機能の中で，COM（n）というのは以下の意味を持ちます．

COM 0は，本体内RS—232 Cポートを示します．

COM 1～COM 4は，拡張ユニット内のRS—232 Cポートを示します．

〔例〕

```
10 OPEN"I",#1,"COM0:(S8N2)"
20 ON COM(0) GOSUB 100
30 COM (0) ON
40 GOTO 40
50 END
100 INPUT#1,A$
110 PRINT A$
120 IF A$<>"END" THEN RETURN
130 CLOSE#1
140 RETURN 50
```

3.8.2 COM(n) ON/OFF/STOP (コム・オン/オフ/ストップ：communication(n) on/off/stop)

〔機　能〕

通信回線からの入力割込みの許可，禁止，停止をします．

〔形　式〕

```
COM(ポート番号) {ON | OFF | STOP}
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ポート番号は0～4です．
・ONにより，ポート番号により指定された通信ポートからの入力割込みを許可します．この文を実行後，指定ポートからの割込み入力があるとON COM(n) GOSUBで指定されたルーチンが呼出されます．
・OFFにより，指定した通信ポートからの入力割込みを禁止します．
・STOPにより，指定した通信ポートからの入力割込みを一時停止します．この後，COM(n) ONが実行されると，ON COM(n) GOSUBで指定されているルーチンに制御が渡ります．

〔例〕

3.8.1 ON COM(n) GOSUB文を参照して下さい．

3.8.3 OPEN (オープン：open)

〔機 能〕

通信回線の入出力を可能にするため，ファイルをオープンします．

〔形 式〕

```
OPEN "モード",[#]ファイル番号,"COMn：
        [(オプション)]"
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・モードは，I（入力）またはO（出力）のいずれかです．
・ファイル番号は，RS−232C の通信ポートに割当てる番号であり，1～16です．
・デバイス名は，COM0：～COM4：のいずれかです．
・オプションは cbps の形式で指定して，かっこで囲み次のことを表わします．

c−クロックを指定します．
  S：slow クロック（1/64）　F：fast クロック（1/16）
b−データのビット長を指定します．
  8：8ビット/文字　7：7ビット/文字
p−パリティの形式，有無を指定します．
  N：パリティなし　O：奇数パリティ　E：偶数パリティ
s−ストップビットのビット数を指定します．
  2：2ストップビット　1：1ストップビット
オプション指定を省略したときは，(S8N2) になります．

〔例〕

```
10 OPEN "O", #1, "COMO：(S8N2)"
```

3.8.4 CLOSE （クローズ：close）

〔機　能〕

通信回線に割当てられたファイルをクローズします．

〔形　式〕

```
CLOSE  [[#]ファイル番号
       [,[#]ファイル番号]…]
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイル番号はOPEN文で指定した数字です．

〔例〕

3.8.1 ON COM(n) GOSUBを参照して下さい．

3.8.5 INPUT＃（インプット・シャープ：input ＃）

〔機　能〕

通信回線からデータを入力します．

〔形　式〕

INPUT＃　ファイル番号,変数名[, 変数名]…

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイル番号で指定されたRS－232C通信ポートからデータを入力し，指定された変数に代入します．
データの区切りは，CR，コンマ（，）またはコロン（：）で，これらの文字コードは，変数に代入されません．

・通信回線からデータを受信すると，割込みが発生してデータが入力バッファに貯えられます．入力バッファの大きさは127バイトですが，バッファ内にINPUT＃，LINE INPUT，I NPUT＄により未だ読込まれてないデータが127バイトあるときに，データを受信するとバッファが一杯なために受信続行が不可能になり，"Buffer Overflow" のエラーになります．
INPUT＃文は，入力バッファから必要な長さのデータをとり出し，指定されている変数に代入します．このとき，入力バッファ内に読みとるべきデータがなくなると，データがバッファに入ってくるまでループ状態になります．したがって，このようなことを避けるためには，LOFまたはEOF関数によりバッファの状態を知ってから，INPUT＃文を実行するようにします．

〔例〕

3.8.1　ON　COM（n）　GOSUB文を参照して下さい．

3.8.6 LINE INPUT＃（ライン・インプット・シャープ：line input ＃）

〔機　能〕

通信回線から1行のデータを入力します．

〔形　式〕

LINE INPUT＃　ファイル番号,文字変数名

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイル番号で指定されたRS－232C通信ポートからデータを入力し，文字変数に代入します．
CRコードが現れるまでのデータを文字変数に読込みますが，255文字を読込んでもCRコードが現れないときは，そこで読込みが終わります．
なお，CRコードは，文字変数に代入されません．

〔例〕

```
10 OPEN"O",#1,"COM0:(S8N2)"
20 OPEN"I",#2,"COM0:(S8N2)"
30 LINEINPUT "DATA",A$
40 PRINT #1,A$
50 LINEINPUT #2,B$
60 PRINT B$
70 IF B$="END" THEN 90
80 GOTO 30
90 CLOSE #1,#2
100 END
```

3.8.7 PRINT＃（プリント・シャープ：print ＃）

〔機　能〕

通信回線にデータを出力します.

〔形　式〕

PRINT＃　ファイル番号[,式[{, ;}式] …]

または

PRINT＃　ファイル番号,USING
　　　　フォーマット文字列;式[{, ;}式] …

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・ファイル番号で指定されたRS−232C通信ポートにデータを出力します.

〔例〕

```
10 OPEN"O",#1,"COM0:(S8N2)"
20 LINEINPUT "DATA";A$
30 IF A$="END" THEN 60
40 PRINT#1,A$
50 GOTO 20
60 CLOSE#1
70 END
```

3.8.8 LIST （リスト：list）

〔機　能〕

通信回線にプログラムリストを出力します．

〔形　式〕

```
LIST "COMn：[(オプション)]"[,[行番号]
     [{－ ,}[行番号]]]
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・COMn：で指定されるRS－232C通信ポートへプログラムリストを出力します．デバイス名として，COM0：～COM4：が使用できます．オプションについてはOPENの項を参照して下さい．

〔例〕

```
LIST"COM0:(S8N2)",20-500
```

3.9 数 値 関 数

3.9.1 ABS（アブソリュート：absolute）

〔機 能〕

絶対値を与えます．

〔形 式〕

```
ABS（式）
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式は数値式です．

〔例〕

```
10 A=-3:B=6
20 PRINT ABS(A)
30 PRINT ABS(B)
40 END

Ready
RUN
 3
 6

Ready
```

3.9.2 ATN （アーク・タンジェント：arc tangent）

〔機　能〕

三角関数アークタンジェントの値を与えます．

〔形　式〕

ATN（式）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式でなければなりません．
・演算結果は－π/2 から π/2 の範囲の値になります．
・ATN 関数の演算は単精度で行われます．

〔例〕

```
10 INPUT A
20 B=ATN(3.14159/180*A)
30 PRINT B

Ready
RUN
? 350
 1.40853

Ready
```

3.9.3 COS (コサイン：cosine)

〔機 能〕

三角関数コサインの値を与えます．

〔形 式〕

COS (式)

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式は数値式でなければなりません．また，その単位はラジアン(π/180×角度)です．

・COS 関数は単精度で行われます．

〔例〕

```
10 INPUT A
20 B=COS(3.14159/180*A)
30 PRINT B

Ready
RUN
? 60
 .5
```

3.9.4 EXP （イクスポネンシャル：exponential）

〔機　能〕

e を底とした指数関数の値を与えます．

〔形　式〕

EXP（式）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式です．その値は 88.0297 未満でなければなりません．
・EXP 関数がオーバーフローした場合は，〝Overflow〞のエラーメッセージが表示されます．
・EXP 関数は，単精度で行われます．

〔例〕

```
10 INPUT A
20 B=EXP(A)
30 PRINT B

Ready
RUN
? 5
 148.413

Ready
```

3.9.5 FIX （フィックス：fix）

〔機 能〕

引数の値の整数部分を与えます．

〔形 式〕

```
FIX（式）
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

- 式は数値式です．
- FIX（X）はSGN（X）＊INT(ABS(X)) と同じです．
- FIX（X）とINT（X）の値は，Xが正のときは同じですが，Xが負のときは違うので注意する必要があります．

〔例〕

```
10 INPUT A
20 B=FIX(A)
30 C=INT(A)
40 PRINT B,C

Ready
RUN
? -65.4
-65              -66

Ready
```

3.9.6 INT（インティジャー：integer）

〔機 能〕

引数の値を超えない最大の整数を与えます．

〔形 式〕

INT（式）

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式は数値式です．

・整数値を与える関数 FIX との違いに注意して下さい．

〔例〕

```
10 INPUT A
20 B=INT(A)
30 C=FIX(A)
40 PRINT B,C

Ready
RUN
? -36.8
-37              -36

Ready
```

3.9.7 LOG (ログ：logarithm)

〔機 能〕

自然対数の値を与えます．

〔形 式〕

LOG (式)

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式は数値式で，その値は正でなければなりません．

・LOG 関数の演算は単精度で行われます．

〔例〕

```
10 INPUT A
20 B=LOG(A)
30 PRINT B

Ready
RUN
? 59
 4.07754

Ready
```

3.9.8 RND （ランド：random）

〔機　能〕

0と1の間の乱数を与えます．

〔形　式〕

```
RND [(式)]
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・プログラムがRUNされる毎に同じ系列の乱数が生成されますが，RANDOMIZE文によりその系列を変えることもできます．（RANDOMIZE文を参照）

・式は数値式であり，発生される乱数は式の値によって次のようになります．

式の値が負のとき，その負値の固有の乱数系列を作ります．

この系列はRANDOMIZEの影響を受けません．

プログラムを実行するごとに別の乱数系列を必要とするときは，式に違った負値を与えなければなりません．

式の値が0のとき，1つ前に発生した乱数を与えます．

式の値が正または省略のとき，同じ乱数系列の次の乱数を与えます．

〔例〕

```
10 FOR I=1 TO 3          Ready
20 PRINT RND(1)          RUN
30 NEXT                   .591065
40 PRINT                  .207991
50 X=RND(-6)              .550967
60 PRINT
70 FOR I=1 TO 3
80 PRINT RND(1)           .208935
90 NEXT                   .707458
100 PRINT                 .353187
110 PRINT RND(0)
120 END                   .353187

                         Ready
```

3.9.9 SGN（サイン：sign）

〔機　能〕

引数が正の場合は1を，0の場合は0を，負の場合は－1を与えます．

〔形　式〕

SGN（式）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式です．

〔例〕

```
10 INPUT A
20 B=SGN(A)
30 PRINT B
Ready
RUN
? -5
-1

Ready
RUN
? 6
 1

Ready
```

3.9.10 SIN （サイン：sine）

〔機　能〕

三角関数サインの値を与えます．

〔形　式〕

SIN（式）

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・式は数値式でなければなりません．また，その単位はラジアン（π/180×角度）です．

・SIN 関数の演算は単精度で行われます．

〔例〕

```
10 INPUT A
20 B=SIN(3.14159/180*A)
30 PRINT B

Ready
RUN
? 60
 .866025

Ready
```

3.9.11 SQR (スクェア・ルート：square root)

〔機 能〕

平方根を与えます.

〔形 式〕

```
SQR (式)
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式は数値式で，その値は0または正でなければなりません.

・SQR 関数の演算は単精度で行われます.

〔例〕

```
10 INPUT A
20 B=SQR(A)
30 PRINT B

Ready
RUN
? 2
 1.41421

Ready
```

3.9.12 TAN (タンジェント：tangent)

〔機 能〕

三角関数タンジェントの値を与えます.

〔形 式〕

TAN (式)

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式は数値式でなければなりません. また, その単位はラジアン(π/180×角度)です.
・TAN 関数の演算は単精度で行われます.

〔例〕

```
10 INPUT A
20 B=TAN(3.14159/180*A)
30 PRINT B

Ready
RUN
? 60
 1.73205

Ready
```

3.9.13 CSNG （コンバート・シングル：convert to single）

〔機 能〕

整数値，倍精度実数値を単精度実数値に変換します．

〔形 式〕

```
CSNG（式）
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式の値を有効数字６桁の単精度実数型に変換します．その値が－1.70141E＋38～1.70141E＋38の範囲にない場合には，〝Overflow〞のエラーになります．

〔例〕

```
10 A#=5649.6517#
20 B=CSNG(A#)
30 PRINT A#,B

Ready
RUN
 5649.6517      5649.65

Ready
```

3.9.14 CDBL （コンバート・ダブル：convert to double）

〔機　能〕

整数値，単精度実数値を倍精度実数値に変換します．

〔形　式〕

CDBL（式）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・CDBL は式の値を倍精度実数型に型変換するだけであり，精度そのものに変化はありません．

〔例〕

```
10 A=4020.12
20 B#=CDBL(A)
30 PRINT A,B#

Ready
RUN
 4020.12         4020.119873046875

Ready
```

3.9.15 CINT （コンバート・インティジャー：convert to integer）

〔機 能〕

単精度実数値，倍精度実数値を整数値に変換します.

〔形 式〕

```
CINT（式）
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・小数部分を四捨五入して整数値に変換します.

・式の値が－32768 から 32767 までの範囲にないときには，〝Overflow〞エラーが起こります.

〔例〕

```
10 A=35.649:B=-5.528
20 C=CINT(A):D=CINT(B)
30 PRINT C,D

Ready
RUN
 36            -6

Ready
```

3.10 ストリング関数

3.10.1 CHR＄ （キャラクター・ダラー：character ＄）

〔機　能〕

引数の値に対応する文字を与えます．

〔形　式〕

CHR＄ （式）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式の値は0～255の範囲でなければなりません．文字コードについては，キャラクタコード表を参照して下さい．

・ASC関数は，CHR＄の逆の関数であり，文字をキャラクタコードに変換します．

〔例〕

```
10 FOR I=&H20 TO &H4F
20 PRINT CHR$(I);
30 NEXT
```

キャラクタコードの&H20から&H4Fまでを出力しています．

```
RUN
 !"#$%&'()*+,-./0123456789:;<=>?@ABCDEFGHIJKLMNO
```

3.10.2 HEX$ （ヘキサ・ダラー：hexa decimal $）

〔機　能〕

整数を16進数の文字列に変換します．

〔形　式〕

HEX$(式)

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式でなければなりません．
　式が小数部分を含む場合は，その値に対してINT関数を内部で実行して整数化した後で変換を行います．
・変換結果は〝0″～〝FFFF″になります．

〔例〕

```
10 INPUT "10 シンスウ ヲ ト゛ウソ゛ ",A
20 PRINT "&H";HEX$(A)

Ready
RUN
10 シンスウ ヲ ト゛ウソ゛ 186
&HBA

Ready
```

10進を数入力すると，16進数に変換して出力します．

3.10.3 LEFT$ （レフト・ダラー：left $）

〔機　能〕

文字列の左から指定された数の文字列を取出します.

〔形　式〕

LEFT$ （文字式，文字数）

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・文字数は0～255の範囲になければなりません. 文字数で示される値が0ならば，関数値は空文字列になります. また文字数の指定が，文字式で示される文字データの文字数より大きい場合，関数値は文字データ全体となります.

〔例〕

```
10 A$="FUJITSU PERSONAL COMPUTER"
20 B$=LEFT$(A$,7)
30 PRINT B$

Ready
RUN
FUJITSU

Ready
```

3.10.4 MID$ （ミッド・ダラー：middle $）

〔形式1〕

MID$ （文字式，文字位置[,文字数]）

〔機　能〕

指定された文字位置から任意の文字数を取出します．

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

- 文字位置は，文字データの初めから数えた取出す位置を示し，1～255の範囲の値でなければなりません．
- 文字数で示される値は0～255の範囲です．
- 文字数が0のとき，および文字位置が文字データの文字数を超えていたときは，関数値は空文字列になります．
  また，文字数が省略されたとき，あるいは文字数が文字データの残りの文字数より大きいときは，関数値は残りの文字列全体になります．

〔例〕

```
10 A$="FUJITSU PERSONAL COMPUTER"
20 B$=MID$(A$,9,8)
30 PRINT B$

Ready
RUN
PERSONAL

Ready
```

〔形式2〕

MID$（文字変数名,文字位置[,文字数]）=文字式

〔機　能〕

代入文の左辺に書かれ，文字変数内の指定された場所を，文字式のデータで指定された文字数分置換えます．

この代入文を実行しても，文字変数の長さは変わりません．

〔バージョン〕　V1.0　V2.0　V3.0

〔例〕

```
10 A$="FUJITSU PERSONAL COMPUTER"
20 MID$(A$,9,8)=" ﾊﾟｰｿﾅﾙ "
30 PRINT A$

Ready
RUN
FUJITSU  ﾊﾟｰｿﾅﾙ  COMPUTER

Ready
```

3.10.5 OCT$ （オクタル・ダラー：octal $）

〔機　能〕

整数を8進数の文字列に変換します．

〔形　式〕

OCT$ （式）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式でなければなりません．

・式が小数部分を含む場合は，その値に対して INT 関数を内部で実行し整数化した後で変換を行います．

〔例〕

```
10 INPUT "8 ｼﾝｽｳ ｦ ﾄﾞｳｿﾞ",A
20 PRINT OCT$(A)
30 GOTO 10

Ready
RUN
8 ｼﾝｽｳ ｦ ﾄﾞｳｿﾞ50
62
8 ｼﾝｽｳ ｦ ﾄﾞｳｿﾞ30
36
8 ｼﾝｽｳ ｦ ﾄﾞｳｿﾞ


Break In 10
Ready
```

10進数を入力すると，8進に変換して出力します．

3.10.6 RIGHT＄ （ライト・ダラー：right ＄）

〔機　能〕

文字列の右から指定された数の文字列を取出します．

〔形　式〕

RIGHT$ （文字式，文字数）

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・文字数は0～255の範囲になければなりません．文字数が0の場合，関数値は空文字列になります．また文字数が文字式で示される文字データの長さよりも大きい場合は，関数値は文字データ全体となります．

〔例〕

```
10 A$="FUJITSU PERSONAL COMPUTER"
20 B$=RIGHT$(A$,8)
30 PRINT A$
40 PRINT B$

Ready
RUN
FUJITSU PERSONAL COMPUTER
COMPUTER

Ready
```

3.10.7 SPACE$ （スペース・ダラー：space $）

〔機 能〕

指定された数の空白よりなる文字列を与えます．

〔形 式〕

```
SPACE$ （個数）
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・個数は0～255の範囲でなければなりません．また，個数が0のときは空文字列になります．

〔例〕

```
10 A$=SPACE$(5)+"PERSONAL COMPUTER"
20 PRINT A$

Ready
RUN
     PERSONAL COMPUTER

Ready
```

3.10.8 STR$ （エス・ティー・アール・ダラー：string $）

〔機　能〕

数値を表わす文字列を与えます．

〔形　式〕

STR$ （式）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・式は数値式でなければなりません．

・STR $ と VRL 関数は，互いに逆の機能を持つ関数です．

〔例〕

```
10 A=5649
20 B$="A="+STR$(A)
30 PRINT B$

Ready
RUN
A= 5649

Ready
```

3.10.9 STRING$ （ストリング・ダラー：string $）

〔機　能〕

指定された文字だけで作られた文字列を与えます．

〔形　式〕

STRING$ （文字数, { キャラクタコード / 文字式 } ）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・文字数およびキャラクタコードは0～255の範囲でなければなりません．文字数が0のときは，空文字列になります．

・文字式の結果が2文字以上のときは，その先頭の文字だけが有効になります．

〔例〕

```
10 A$=STRING$(20,"A")
20 B$=STRING$(10,"B")
30 PRINT A$
40 PRINT B$

Ready
RUN
AAAAAAAAAAAAAAAAAAAA
BBBBBBBBBB

Ready
```

3.10.10 ASC （アスキー：ascii）

〔機　能〕

指定した文字列の最初の文字のキャラクタコードを与えます.

〔形　式〕

ASC （文字式）

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ASC関数の結果は，文字列の最初の文字のキャラクタコードを表わす数値です.

・文字式の結果が空文字列のときは，"Illegal Function Call" のエラーになります.

・CHR $ 関数は，ASCの逆の関数であり，キャラクタコードから文字に変換するときに使われます.

〔例〕

```
10 A$="A"
20 A=ASC(A$)
30 PRINT A

Ready
RUN
 65

Ready
```

3.10.11 INSTR （インストリング：instring）

〔機　能〕

指定された文字列を検索し，見つかった位置を与えます．

〔形　式〕

INSTR （［検索開始位置,］文字式1，文字式2）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・文字式1で示される文字データの中から文字式2のデータを探し，その位置を与えます．検索開始位置が省略されたときは，文字式1のデータの先頭から探し始めます．

・検索開始位置が，文字式1の文字数より大きいとき，文字式1が空文字列のとき，文字式2が見つからなかった場合には，関数値は0になります．

・文字式2が空文字列の場合には，関数値は，検索開始位置または1になります．

・検索開始位置は1から255の範囲で指定しますが，その範囲にないときは "Illegal Function Call" のエラーになります．

〔例〕

```
10 A$="FUJITSU"
20 A=INSTR(1,A$,"I")
30 PRINT A

Ready
RUN
 4

Ready
```

3.10.12 LEN （レングス：length）

〔機　能〕

文字列の長さを与えます．

〔形　式〕

LEN （文字式）

〔説　明〕

・文字式の結果が空文字列のときは0になります．

〔バージョン〕　V1.0　V2.0　V3.0

〔例〕

```
10 A$="FUJITSU PERSONAL COMPUTER"
20 A=LEN(A$)
30 PRINT A

Ready
RUN
 25

Ready
```

3.10.13 VAL （バリュー：value）

〔機　能〕

文字列の表わす数値を与えます．

〔形　式〕

VAL （文字式）

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・文字列の最初の文字が＋，－，&，・，または数字でないときは，関数値は0になります．数字以外の文字（16進数の場合は，0～A～F以外の文字，8進数の場合は0～7以外の文字）が現われた場合は，それ以降の文字は無視されます．

・文字列の途中の空白は読みとばされます．

・STR＄関数は，VALとは逆の機能を持つ関数であり，数値を文字列に変換します．（STR＄を参照）

〔例〕

```
10 A$="6517"
20 A=VAL(A$)
30 B=VAL("&H"+A$)
40 PRINT A
50 PRINT B

Ready
RUN
 6517
 25879

Ready
```

3.11 一　般　関　数

3.11.1 CSRLIN（カーソル・ライン：cursor line）

〔機　能〕

画面上のカーソルの垂直位置を与えます.

〔形　式〕

```
CSRLIN
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・画面上のカーソルの垂直位置を行単位で与えます.

〔例〕

```
10 LOCATE 20,5
20 A=CSRLIN
30 B=POS(0)
40 LOCATE 5,7
50 PRINT A,B


RUN






          5           20

Ready
```

3.11.2 POS (ポジション：position)

〔機 能〕

カーソル，プリンタヘッドの水平位置を与えます．

〔形 式〕

```
POS (ファイル番号)
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・ファイル番号が0のときは，画面上のカーソルの水平位置を与えます．

・ファイル番号がプリンタに割当てられているときは，プリンタバッファ内のプリンタヘッドの現在の位置を与えます．

〔例〕

3.11.1 CSRLIN文を参照して下さい．

3.11.3 LPOS（ラインポジション：line position）

〔機　能〕

プリンタヘッドの水平位置を与える関数です．

〔形　式〕

```
LPOS（式）
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

式は意味を持ちませんが，形式として記述します．

〔例〕

```
10 FOR I=&H20 TO &H3F
20 LPRINT CHR$(I);
30 PRINT LPOS(0);
40 NEXT I

Ready
RUN
 1  2  3  4  5  6  7  8  9  10  11  12
 13  14  15  16  17  18  19  20  21  22
 23  24  25  26  27  28  29  30  31  32

Ready
```

PRINTER 出力

```
 !"#$%&'()*+,-./0123456789:;<=>?
```

3.11.4 POINT (ポイント：point)

〔機　能〕

指定した位置にドットがセットされているかいないかを与えます.

〔形　式〕

POINT (水平位置, 垂直位置)

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・指定したグラフィック座標にドットがセットされている場合には－1, セットされていない場合(背景色で表示されていたとき)は0を与えます.

〔例〕

```
10 PSET(320,100,7)
20 A=POINT(320,100)
30 B=POINT(321,100)
40 PRINT A,B
```

3.11.5 ERR/ERL (エラー・ナンバー/エラー・ライン : error number/error line)

〔機 能〕

発生したエラーのエラー番号及び行番号を与えます.

〔形 式〕

ERR/ERL

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・エラーが発生すると, システム変数ERRはエラーコードを, ERLはエラーの起った行番号を保持しています. エラーが直接モードで実行している文で起きたときには, ERLは行番号として65535を持っています.

・一般にERRとERLは, ON ERROR GO TOにより指定されるエラー処理ルーチン内で, プログラムの流れの制御のために使用されます.
エラールーチン内でRESUMEが実行されると, ERRは0になりますが, ERLは次にエラーが発生するまでは, 直前のエラー行番号が保持されています.

〔例〕

3.2.27 ON ERROR GOTO 文を参照して下さい.

3.11.6 VARPTR （バリアブル・ポインター：variable pointer）

〔機　能〕

変数名で指定されるデータの格納されている先頭番地を与えます．

〔形　式〕

```
VARPTR （変数名）
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・変数名は任意の型（数値，文字，配列）を使うことができますが，VARPTR を使う前にその変数には値が代入されてなければなりません．そうでないときには，"Illegal Function Call" のエラーになります．

・配列に対して VARPTR を使用するときには，全ての単純変数に値が代入されていなければなりません．これは，新しい変数に値が代入されると，配列のアドレスが変わるためです．

・VARPTR の値の示す番地には，データが次のように格納されています．

① 整数型のとき

0	上位8ビット
1	下位8ビット

② 単精度実数型のとき

③ 倍精度実数型のとき

④ 文字型のとき

0	文字列の長さ
1 2	文字列が格納されているアドレス

〔例〕

```
10 A$="FUJITSU"
20 B=VARPTR(A$)
30 PRINT HEX$(B)

Ready
RUN
CC0

Ready
```

3.11.7 USR （ユーザ：user）

〔機　能〕

引数を持ってユーザのアセンブリ言語ルーチンを呼出します．

〔形　式〕

USR ［数字］（引数）

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・数字は0から9までの整数で，DEF USR文で定義した番号に対応します．
・数字を省略すると，USR0と見なされます．

〔例〕

3.2.2 DEF USR文を参照して下さい．

3.11.8 PEEK (ピーク：peek)

〔機 能〕

メモリの内容を読出します.

〔形 式〕

PEEK (式)

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・式の値は0から65535までの範囲の数であり，読出すメモリアドレスを示します.

・PEEKはPOKEの逆の働きをする関数です.

〔例〕

```
10 POKE &H5000,&H39
20 B=PEEK(&H5000)
30 PRINT B

Ready
RUN
 57

Ready
```

3.11.9 FRE (フリー：free)

〔機　能〕

引数が数値の場合はBASICが使っていないメモリのバイト数を，引数が文字の場合は文字領域の未使用バイト数を与えます．

〔形　式〕

```
FRE (整数)
FRE (文字列)
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・引数に文字列が指定されたときは，文字領域の未使用バイト数を求める前に文字領域の整理を行います．このことは，一般に〝ガーベッジコレクション〟と呼ばれますが，文字領域の中の不要な文字列を消去して，必要な文字列だけを残すことです．

〔例〕

```
PRINT FRE(0)
 25559

Ready
PRINT FRE("A")
 300

Ready
```

3.11.10 TIME$ （タイム・ダラー：time $）

〔形式 1〕

```
TIME$
```

〔機　能〕

内蔵のタイマの示す時刻を与えます．

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・TIME$ の形式は，

hh：mm：ss

で示され，それぞれ時，分，秒を表わします．hh は 00 から 23 まで，mm は 00 から 59 まで，ss は 00 から 59 までの数です．電源を投入した直後に，タイマは全て 00：00：00 にセットされます．

・TIME$ への代入は次の形式で行います．

TIME$＝"hh：mm：ss"

〔例〕

```
10 WIDTH 40,25
20 INPUT "HH:MM:SS";T$
30 TIME$=T$
40 LOCATE 0,10
50 PRINT TIME$
60 GOTO 40

HH:MM:SS? "11:00:00"




11:00:07


Break in 50
Ready
```

〔形式 2〕

```
TIME
```

〔バージョン〕 (V1.0) (V2.0) (V3.0)

〔説 明〕

・00：00：00 を基準とする秒単位の時刻を示すシステム変数です．

〔例〕

```
10 WIDTH 40,25
20 LOCATE 0,5
30 PRINT TIME
40 GO TO 20


 39640


Break In 30
Ready
```

3.11.11 DATE$ （デート・ダラー：date $）

〔形式1〕

```
DATE$
```

〔機　能〕

内蔵タイマの示す日付を表わします．

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・日付は次の形式です．

yy/mm/dd

yy が年，mm が月，dd が日を表わし，それぞれ2桁の数字です．

・DATE$ への代入は次の形式で行います．

DATE$ = "yy/mm/dd"

yy ：西暦 1981→81

〔例〕

```
10 WIDTH 40,25
20 DATE$="81/10/05"
30 TIME$="23:59:57"
40 LOCATE 0,5:PRINT DATE$,TIME$
50 GOTO 40





81/10/05        23:59:59


Break In 40
Ready
```

〔形式 2〕

```
DATE
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・1月1日を基準とするトータル日数を示すシステム変数です．

〔例〕

```
10 DATE$="81/10/07"
20 PRINT DATE

Ready
RUN
 280

Ready
```

3.11.12 SPC (エス・ピー・シー：space)

〔機 能〕

指定された数の空白をプリントします．

〔形 式〕

```
SPC （個数）
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・空白の個数は0から255まででなければなりません．

・SPC関数はPRINT，LPRINT，およびPRINT＃文中でのみ使うことができます．

〔例〕

```
10 PRINT "FUJITSU"SPC(3)"PERSONAL"SPC(4)
"COMPUTER"

Ready
RUN
FUJITSU   PERSONAL    COMPUTER

Ready
```

3.11.13　TAB　（タブ：tab）

〔機　能〕

指定された桁位置まで空白をプリントします．

〔形　式〕

TAB　（桁位置）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・桁位置は0から32767まで指定できます．桁位置が1行の表示文字数を超えた場合は，1行の表示文字数で割った余りが用いられます．また，その値が現在の表示位置よりも小さければ，次の行において同様のことが行われます．

・TAB関数はPRINT，LPRINTおよび，PRINT＃文中で用いられます．

〔例〕

```
10 INPUT "NAME,TEL";A$,B$
20 PRINT "NAME"TAB(20)"TEL"
30 PRINT A$;TAB(20);B$
```

3.11.14 SCREEN（スクリーン：screen）

〔機　能〕

指定座標のキャラクタコード，および属性を与えます．

〔形　式〕

```
SCREEN （水平位置，垂直位置
        [,セレクトスイッチ]）
```

〔バージョン〕 V1.0　V2.0　V3.0

〔説　明〕

・水平位置，垂直位置は，読取ろうとする画面上のキャラクタ座標です．

・セレクトスイッチは，0または1であり，次のことを示します．

セレクトスイッチが省略されたとき，および0のときは，指定座標のキャラクタコードが返されます．

セレクトスイッチが1のときは，指定座標の属性が返されます．属性は，その値を8で割った余りがパレットコードを示し，8で割った商が0ならばノーマル表示，1ならばリバース表示を示します．なお，指定した座標にキャラクタの表示がないときは，いずれの場合も0が返されます．

〔例〕

```
10 CLS
20 LOCATE 0,0
30 PRINT "FUJITSU PERSONAL COMPUTER"
40 LOCATE 0,5
50 FOR I=24 TO 0 STEP -1
60 A=SCREEN(I,0)
70 PRINT CHR$(A);
80 NEXT

FUJITSU PERSONAL COMPUTER




RETUPMOC LANOSREP USTIJUF
Ready
```

3.12 入出力関数

3.12.1 CVI/CVS/CVD

(シー・ブイ・アイ：convert to integer
シー・ブイ・エス：convert to single
シー・ブイ・ディ：convert to double)

(ディスク)

〔機能〕

文字で表現された値を実際の数値データに変換します．

〔形式〕

CVI　(2バイトの文字列) CVS　(4バイトの文字列) CVD　(8バイトの文字列)

〔バージョン〕 V1.0 V2.0 V3.0

〔説明〕

・ランダムファイルから読込んだデータは，すべて文字型になっているため，数値として扱うためには，それを数値データに変換する必要があります．
CVI，CVS，CVDは文字を数値データに変換するときに用います．

・CVIは2バイトの文字列を整数に，CVSは4バイトの文字列を単精度形式数値に，CVDは8バイトの文字列を倍精度形式の数値にそれぞれ変換します．

〔例〕

```
10 OPEN "R",#1,"TESTDATA"
20 FIELD #1,2 AS I$,4 AS S$,8 AS D$
30 INPUT "DATA ヲ ニュウリョクシテクダサイ";I%,S,D#
40 LSET I$=MKI$(I%)
50 LSET S$=MKS$(S)
60 LSET D$=MKD$(D#)
70 PUT #1,1
80 PRINT
90 GET #1,1
100 I%=CVI(I$)
110 S=CVS(S$)
120 D#=CVD(D$)
130 PRINT I%,S,D#
140 CLOSE #1
150 END

RUN
DATA ヲ ニュウリョクシテクダサイ? 2,32.12,63.25874568

 2              32.12              63.25874568
```

3.12.2 MKI$/MKS$/MKD$

(ディスク)

(エム・ケー・アイ・ダラー：make integer $
エム・ケー・エス・ダラー：make single $
エム・ケー・ディ・ダラー：make double $)

〔機　能〕

数値を文字に変換します.

〔形　式〕

MKI $	(整数表記)
MKS$	(単精度表記)
MKD$	(倍精度表記)

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ランダムファイルバッファに入れる数値は全て文字に変換する必要がありますが，これらの関数は，その変換のために使われます.

・数値から文字への変換は，内部表現された数値をそのまま文字コードとして扱うことによって行います.

・MKI $ は整数を2バイトの文字列にMKS $ は単精度形式の数値を4バイトの文字列に，MKD $ は倍精度形式の数値を8バイトの文字列にそれぞれ変換します.

〔例〕

3.12.1 CVI/CVS/CVD 文を参照して下さい.

3.12.3 EOF（エンド　オブ　ファイル：end of file）

〔機　能〕

ファイルの終わりを検出します．

〔形　式〕

EOF　（ファイル番号）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ファイル番号で示される入力ファイルのEOF（End of File）を検出します．EOFを検出すると－1（真）を，EOFでないときは0（偽）を与えます．

・ファイル番号で示されるファイルは，ディスク，カセット又はRS-232C上のシーケンシャルファイルでなければなりません．

・指定されたファイル番号がRS－232Cの通信ポートに割当てられているときには，そのバッファが空であるか否かを調べ，空でないならば0を，空ならば－1を与えます．

・なお，キーボード（"KYBD："）に割当てられたファイル番号に対して，EOF関数を用いることはできません．

〔例〕

DATA2というシーケンシャルファイルがすでに存在している場合の例です．

```
10 OPEN "I",#1,"DATA2"
20 INPUT #1,A$
30 PRINT A$
40 IF EOF(1) THEN 50 ELSE 20
50 CLOSE #1
60 END

Ready
RUN
822145
822101
235
63254
123485
125468
3256
214
3256
7896

Ready
```

3.12.4 LOF (エル・オー・エフ：length of diskfile)

〔機 能〕

入力バッファ中の文字数を与えます.

〔形 式〕

LOF (ファイル番号)

〔バージョン〕 V1.0 V2.0 V3.0

〔説 明〕

・ファイル番号で示される入力ファイルのバッファに入っている文字数を与えます.
・キーボード("KYBD:")に割当てられたファイル番号に対して, LOF 関数を用いることはできません.
・ランダムファイルに対して, LOF 関数を用いたとき, ランダムファイルの最大レコード番号が返されます.

〔例〕

```
10 OPEN "R",#1,"TESTDATA"
20 FIELD #1,2 AS I$,4 AS S$,8 AS D$
30 GET #1,1
40 A=LOF(1)
50 PRINT A
60 CLOSE
70 END
```

3.12.5 LOC （エル・オー・シー：location counter of diskfile） （ディスク）

〔機　能〕

ランダムファイルに対して次にGETまたはPUTされるレコード番号を与えます．

〔形　式〕

```
LOC （ファイル番号）
```

〔バージョン〕 V1.0 V2.0 V3.0

〔説　明〕

・ファイル番号で示されるファイルは，ディスク上のファイルであり，ランダム入出力モード（"R"）でオープンされていなければなりません．

・ランダムファイル上で，直前にGETまたはPUTされたレコードの次のレコード番号を与えます．レコード番号を伴わないGET文またはPUT文が実行されたときとられるレコード番号と同じになります．

〔例〕

```
10 OPEN"R",#1,"TESTDATA"
20 FIELD#1,2 AS I$,4 AS S$,8 AS D$
30 PUT#1
40 A=LOC(1)
50 PRINT A
60 CLOSE#1
70 END

Ready
RUN
 2
```

3.12.6 DSKI$ （ディスク・アイ・ダラー：disk initialize $） （ディスク）

［機 能］

指定されたセクタの内容をシステムランダムバッファに読込みます．

［形 式］

DSKI$ （ドライブ番号，トラック番号，セクタ番号）

［バージョン］ V1.0 V2.0 V3.0

［説 明］

・この関数を使用する前にFIELD文により，システムランダムバッファに割当てる文字変数の定義をしておかなければなりません．
・ドライブ番号は，システムの構成により0から3，または0から7まで指定できます．
・トラック番号はミニフロッピィディスクのときは0から39，標準フロッピィディスクのときは0～76まで指定できます．
・セクタ番号は，ミニフロッピィディスクのときは1から32まで指定できます．17～32を指定したときは，ディスクの裏面のセクタ1～16を示します．また，標準フロッピィディスクのときは1から52まで指定できます．27～52を指定したときは，ディスクの裏面のセクタ1～26を示します．
・システムランダムバッファはシステムに持つ256バイト領域のことです．

［例］

```
10 FIELD #0,128 AS A$,128 AS B$
20 DUMMY$=DSKI$(0,10,1)
```

ドライブ0，トラック10，セクタ1の内容を文字変数DUMMY$に与えます．

3.12.7 DSKF（ディスク・エフ：diskfile）　（ディスク）

〔機　能〕

ディスクの未使用領域のクラスタ数を示します.

〔形　式〕

```
DSKF （ドライブ番号）
```

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・ドライブ番号は0から3までです.

〔例〕

```
10 A=DSKF (0)
20 PRINT A
30 END

Ready
RUN
 93
```

3.12.8 INPUT$ （インプット・ダラー：input $）

〔機　能〕

キーボードあるいはファイルから指定された文字数の文字列を入力します．

〔形　式〕

INPUT$ （文字数[,[#]ファイル番号]）

〔バージョン〕 (V1.0) (V2.0) (V3.0)

〔説　明〕

・文字数は0から255の範囲でなければなりません．

・ファイル番号が指定されたときは，そのファイルから入力し，ファイル番号の指定のないときは，キーボードから入力します．キーボードからの入力は，INPUT文と異なり，その文字は画面に表示されません．

・キーボード及びRS-232C通信ポートからの入力の場合，そのバッファ内に指定された文字数分の文字があるときは，バッファの先頭からその文字数分を取り出します．またバッファが空のとき，及びバッファ内に必要な文字数分がないときは，指定された文字数の文字の入力があるまで待ち続けます．

・INPUT $ は，BREAKキー（(V1.0)(V2.0)ではSTOPキー）以外のコントロール文字もすべてそのまま読みとるため，INPUT文やLINE INPUT文では読みとれないCRやLFコードもそのまま読みとることができます．

〔例〕

```
10 A$=INPUT$(3)
20 PRINT A$
```

キーボードから，3文字入力し，入力されたデータを出力します．

```
Ready
RUN
123

Ready
```

3.12.9 INKEY$（インキー・ダラー：input key $）

〔機　能〕

キーボードが押されていれば，その文字を与えます．押されていなければ空文字を与えます．

〔形　式〕

```
INKEY$
```

〔バージョン〕　(V1.0)　(V2.0)　(V3.0)

〔説　明〕

・キーボードバッファが空のときには，空文字列が返されます．キー入力によりキーボードバッファが空でないときには，バッファの先頭から1文字を取り出し，その文字を返します．
INKEY $ は BREAK キー（(V1.0) (V2.0) では STOP キー）以外のコントロール文字もすべて読取りますが，読取られた文字は画面に表示されません．

〔例〕

キーボードから入力された文字のアスキーコードを出力します．

```
10 A$=INKEY$
20 IF A$="" THEN 10
30 PRINT A$;"=";HEX$(ASC(A$))
40 GOTO 10

Ready
RUN
1=31
3=33
2=32
5=35
R=52
K=4B
J=4A
G=47


Break In 10
Ready
```

3.12.10 ANPORT（アンポート：analog port）

〔機　能〕

アナログポートからA/D変換されたデータを読取ります.

〔形　式〕

ANPORT（チャンネル番号，電圧レンジ）

〔バージョン〕　V1.0　V2.0　V3.0

〔説　明〕

・A/D変換された0～255の値を与えます. チャンネル番号は0～3で指定し，選択するチャンネルを示します. 電圧レンジは0または1で，入力する電圧範囲を指定し，0のときは0～5/8V，1のときは0～2.5Vを示します.

〔例〕

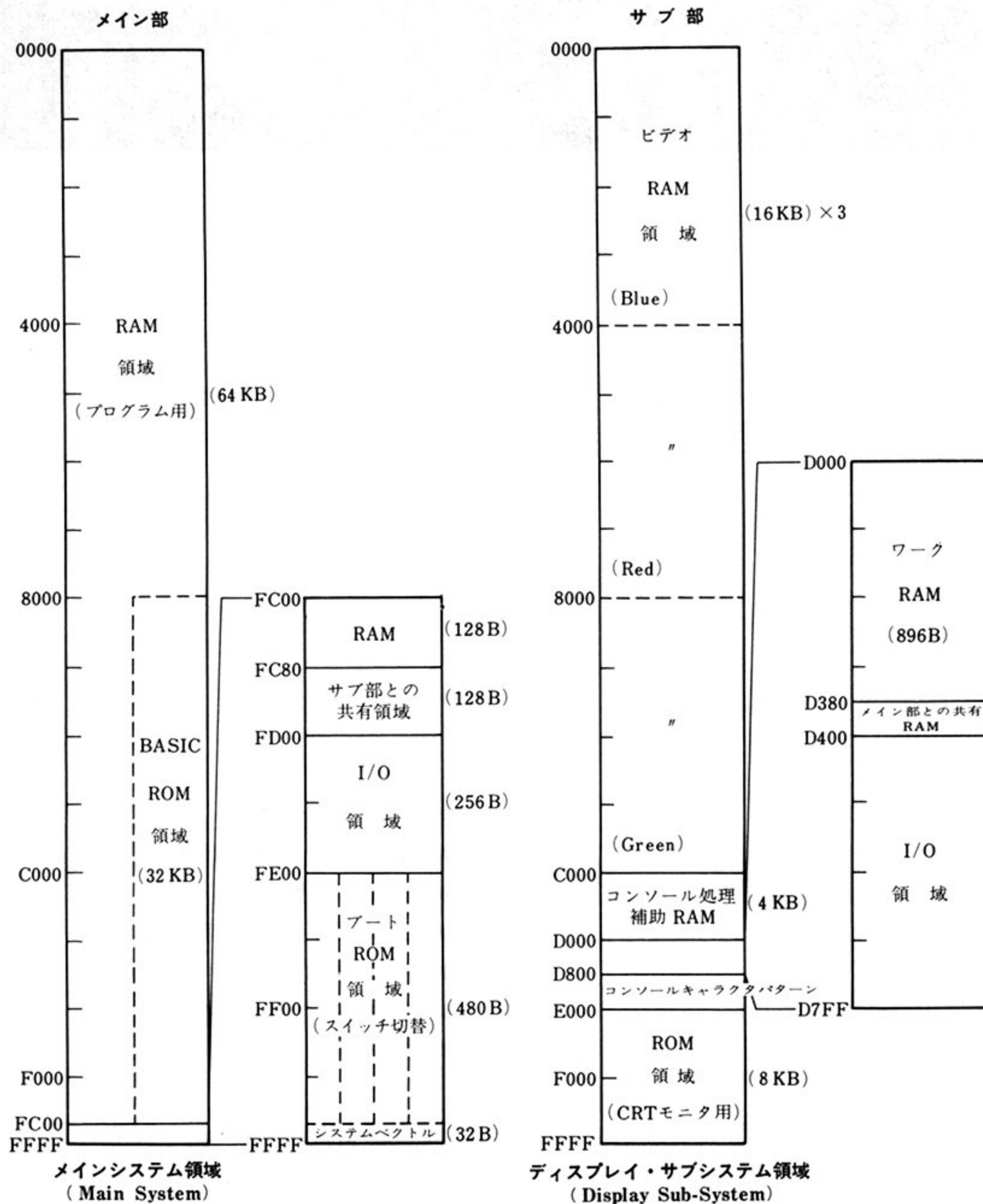
```
10 CLS
20 A=ANPORT(0,1)
30 B=A/3.2
40 A$=STRING$(B,"█")
50 LOCATE 0,5:PRINT A
60 LOCATE 0,7:PRINT A$
70 GOTO 20
80 END
```

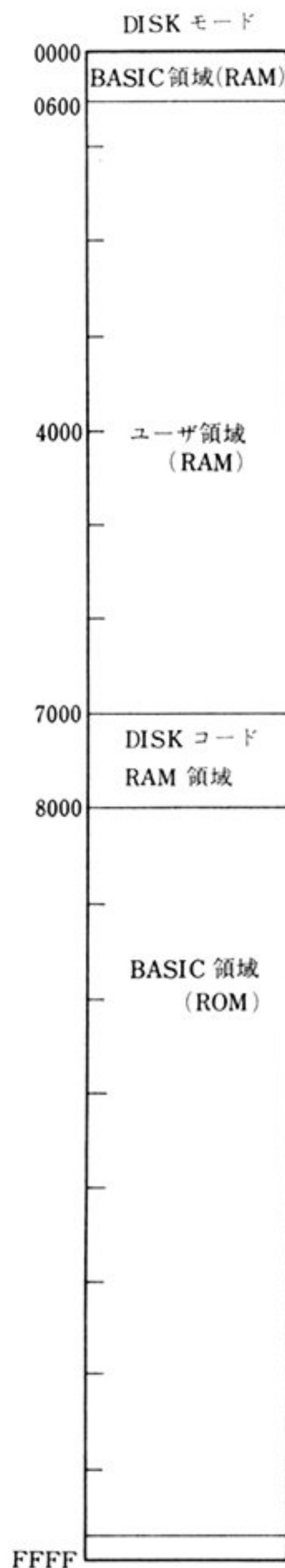
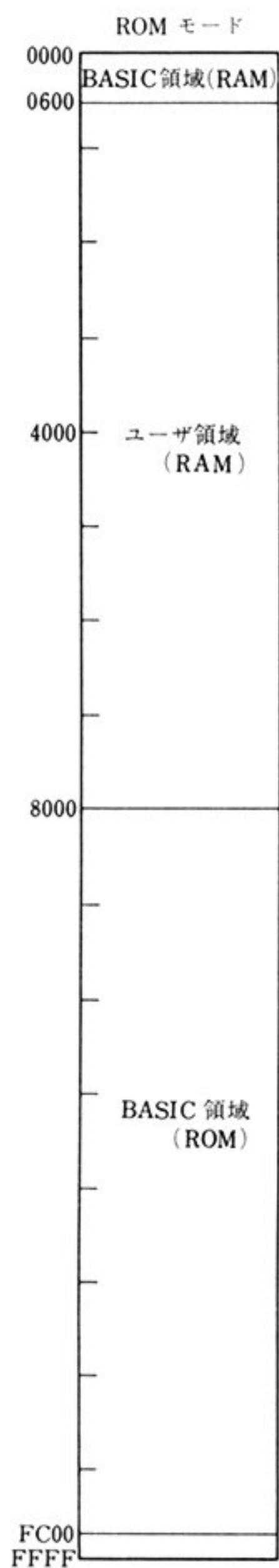
```
 255
```

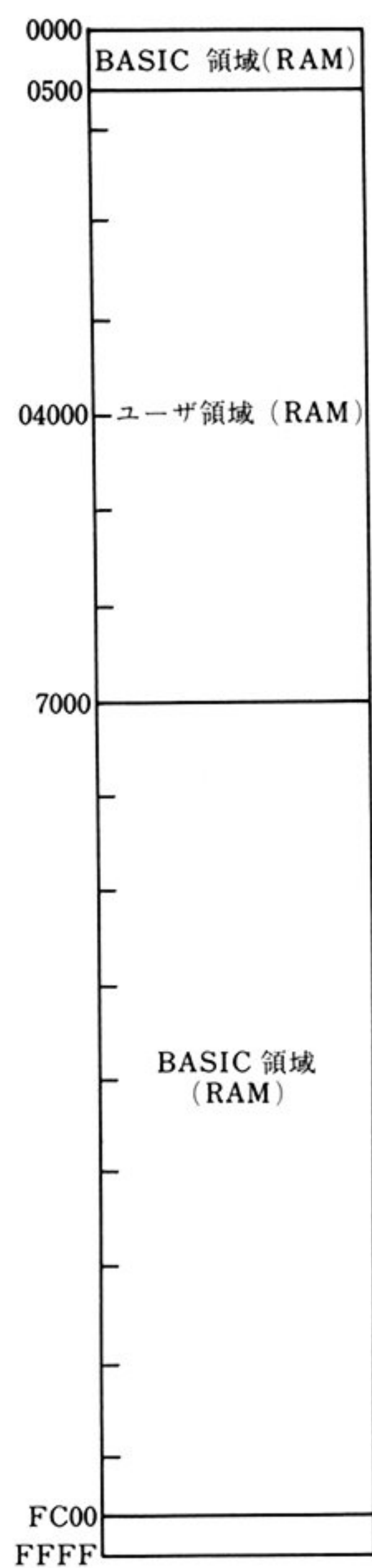
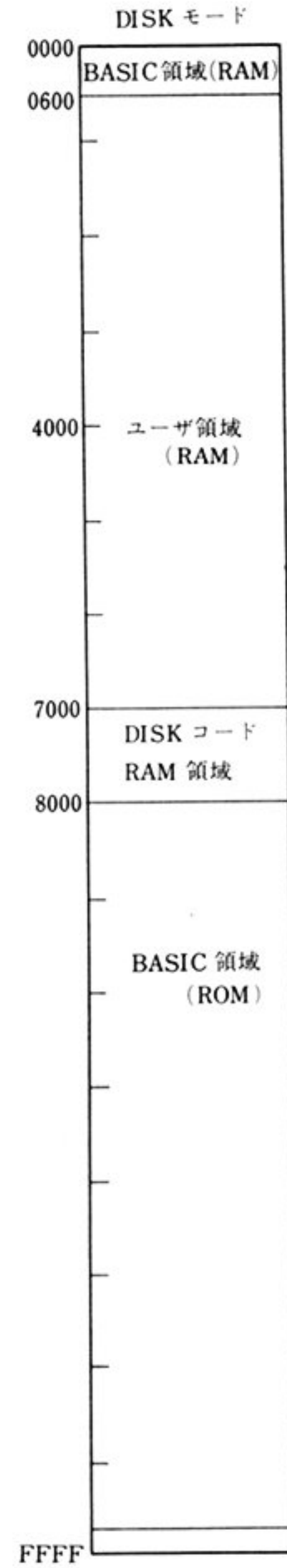
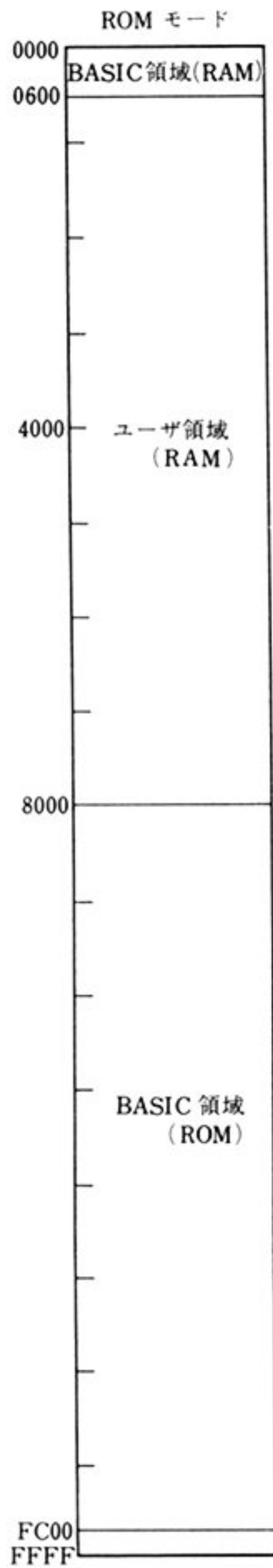
```
Break In 60
Ready
```

付　録

- メモリマップ
- キャラクタコード表
- 非漢字一覧表
- JIS 第 1 水準漢字一覧表
- F-BASIC のエラーメッセージ
- F-BASIC 命令一覧表
- キーボード図

付録1　メモリマップ

メモリマップ

F-BASIC V1.0実行時におけるメイン部のメモリマップは次のようになります.

DISKコードはシステムの起動時に, システムディスクからRAMにロードされます.

F-BASIC V2.0実行時におけるメイン部のメモリマップは次のようになります.

F-BASIC V3.0実行時におけるメイン部のメモリマップは次のようになります.

DISK コードはシステムの起動時に, システムディスクから RAM にロードされます.

付録2　キャラクタコード表

下位＼上位	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
0		DE	(space)	0	@	P	`	p		┴		ー	タ	ミ		╳
1	SH	D1	!	1	A	Q	a	q		┬	。	ア	チ	ム		円
2	SX	D2	"	2	B	R	b	r		┤	「	イ	ツ	メ		年
3	EX	D3	#	3	C	S	c	s		├	」	ウ	テ	モ		月
4	ET	D4	$	4	D	T	d	t			、	エ	ト	ヤ		日
5	EQ	NK	%	5	E	U	e	u		─	・	オ	ナ	ユ		時
6	AK	SN	&	6	F	V	f	v		│	ヲ	カ	ニ	ヨ		分
7	BL	EB	'	7	G	W	g	w			ァ	キ	ヌ	ラ		秒
8	BS	CN	(	8	H	X	h	x		┌	ィ	ク	ネ	リ	♠	〒
9	HT	EM	)	9	I	Y	i	y		┐	ゥ	ケ	ノ	ル	♥	市
A	LF	SB	*	:	J	Z	j	z		└	ェ	コ	ハ	レ	♦	区
B	HM	EC	+	;	K	[	k	¦		┘	ォ	サ	ヒ	ロ	♣	町
C	CL	→	,	<	L	¥	l	|		╭	ャ	シ	フ	ワ	●	村
D	CR	←	−	=	M	]	m	¦		╮	ュ	ス	ヘ	ン	○	人
E	SO	↑	.	>	N	∧	n	‾		╰	ョ	セ	ホ	゛	╱	
F	SI	↓	/	?	O	—	o	DL	┼	╯	ッ	ソ	マ		╲	

付録 3　非漢字一覧表

コードはすべて16進形式

記号

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
212X			、	。	，	．	・	：	；	？	！	゛	゜	´	｀	¨
213X	＾	￣	＿	ヽ	ヾ	ゝ	ゞ	〃	仝	々	〆	〇	ー	―	‐	／
214X	＼	～	∥	｜	…	‥	‘	’	“	”	（	）	〔	〕	［	］
215X	｛	｝	〈	〉	《	》	「	」	『	』	【	】	＋	－	±	×
216X	÷	＝	≠	＜	＞	≦	≧	∞	∴	♂	♀	°	′	″	℃	￥
217X	＄	¢	£	％	＃	＆	＊	＠	§	☆	★	○	●	◎	◇	
222X		◆	□	■	△	▲	▽	▼	※	〒	→	←	↑	↓	〓	

（例えば，％のコードは2173と読みます．実際の使用には“&H”をつけて，「&H2173」とします）

英・数字

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
233X	０	１	２	３	４	５	６	７	８	９						
234X		Ａ	Ｂ	Ｃ	Ｄ	Ｅ	Ｆ	Ｇ	Ｈ	Ｉ	Ｊ	Ｋ	Ｌ	Ｍ	Ｎ	Ｏ
235X	Ｐ	Ｑ	Ｒ	Ｓ	Ｔ	Ｕ	Ｖ	Ｗ	Ｘ	Ｙ	Ｚ					
236X		ａ	ｂ	ｃ	ｄ	ｅ	ｆ	ｇ	ｈ	ｉ	ｊ	ｋ	ｌ	ｍ	ｎ	ｏ
237X	ｐ	ｑ	ｒ	ｓ	ｔ	ｕ	ｖ	ｗ	ｘ	ｙ	ｚ					

ひらがな

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
242X		ぁ	あ	ぃ	い	ぅ	う	ぇ	え	ぉ	お	か	が	き	ぎ	く
243X	ぐ	け	げ	こ	ご	さ	ざ	し	じ	す	ず	せ	ぜ	そ	ぞ	た
244X	だ	ち	ぢ	っ	つ	づ	て	で	と	ど	な	に	ぬ	ね	の	は
245X	ば	ぱ	ひ	び	ぴ	ふ	ぶ	ぷ	へ	べ	ぺ	ほ	ぼ	ぽ	ま	み
246X	む	め	も	ゃ	や	ゅ	ゆ	ょ	よ	ら	り	る	れ	ろ	ゎ	わ
247X	ゐ	ゑ	を	ん												

カタカナ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
252X		ァ	ア	ィ	イ	ゥ	ウ	ェ	エ	ォ	オ	カ	ガ	キ	ギ	ク
253X	グ	ケ	ゲ	コ	ゴ	サ	ザ	シ	ジ	ス	ズ	セ	ゼ	ソ	ゾ	タ
254X	ダ	チ	ヂ	ッ	ツ	ヅ	テ	デ	ト	ド	ナ	ニ	ヌ	ネ	ノ	ハ
255X	バ	パ	ヒ	ビ	ピ	フ	ブ	プ	ヘ	ベ	ペ	ホ	ボ	ポ	マ	ミ
256X	ム	メ	モ	ャ	ヤ	ュ	ユ	ョ	ヨ	ラ	リ	ル	レ	ロ	ヮ	ワ
257X	ヰ	ヱ	ヲ	ン	ヴ	ヵ	ヶ									

（例えば，ケのコードは2531と読みます．実際の使用には“&H”をつけて，「&H2531」とします．）

ギリシャ文字

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
262X		Α	Β	Γ	Δ	Ε	Ζ	Η	Θ	Ι	Κ	Λ	Μ	Ν	Ξ	Ο
263X	Π	Ρ	Σ	Τ	Υ	Φ	Χ	Ψ	Ω							
264X		α	β	γ	δ	ε	ζ	η	θ	ι	κ	λ	μ	ν	ξ	ο
265X	π	ρ	σ	τ	υ	φ	χ	ψ	ω							

ロシア文字

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
272X		А	Б	В	Г	Д	Е	Ё	Ж	З	И	Й	К	Л	М	Н
273X	О	П	Р	С	Т	У	Ф	Х	Ц	Ч	Ш	Щ	Ъ	Ы	Ь	Э
274X	Ю	Я														
275X		а	б	в	г	д	е	ё	ж	з	и	й	к	л	м	н
276X	о	п	р	с	т	у	ф	х	ц	ч	ш	щ	ъ	ы	ь	э
277X	ю	я														

付録 4　JIS 第 1 水準漢字一覧表

コードはすべて 16 進形式

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ア	302X		亜	唖	娃	阿	哀	愛	挨	姶	逢	葵	茜	穐	悪	握	渥
	303X	旭	葦	芦	鯵	梓	圧	斡	扱	宛	姐	虻	飴	絢	綾	鮎	或
	304X	粟	袷	安	庵	按	暗	案	闇	鞍	杏						

（例えば，安のコードは3042と読みます．実際の使用には “&H” をつけて，「&H3042」とします）

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
イ	304X											以	伊	位	依	偉	囲
	305X	夷	委	威	尉	惟	意	慰	易	椅	為	畏	異	移	維	緯	胃
	306X	萎	衣	謂	違	遺	医	井	亥	域	育	郁	磯	一	壱	溢	逸
	307X	稲	茨	芋	鰯	允	印	咽	員	因	姻	引	飲	淫	胤	蔭	
	312X		院	陰	隠	韻	吋										

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ウ	312X							右	宇	烏	羽	迂	雨	卯	鵜	窺	丑
	313X	碓	臼	渦	嘘	唄	欝	蔚	鰻	姥	厩	浦	瓜	閏	噂	云	運
	314X	雲															

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
エ	314X		荏	餌	叡	営	嬰	影	映	曳	栄	永	泳	洩	瑛	盈	穎
	315X	頴	英	衛	詠	鋭	液	疫	益	駅	悦	謁	越	閲	榎	厭	円
	316X	園	堰	奄	宴	延	怨	掩	援	沿	演	炎	焔	煙	燕	猿	縁
	317X	艶	苑	薗	遠	鉛	鴛	塩									

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
オ	317X								於	汚	甥	凹	央	奥	往	応	
	322X		押	旺	横	欧	殴	王	翁	襖	鴬	鴎	黄	岡	沖	荻	億
	323X	屋	憶	臆	桶	牡	乙	俺	卸	恩	温	穏	音				

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
カ	323X													下	化	仮	何

次頁につづく

コードはすべて16進形式

カ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
324X	伽	価	佳	加	可	嘉	夏	嫁	家	寡	科	暇	果	架	歌	河
325X	火	珂	禍	禾	稼	箇	花	苛	茄	荷	華	菓	蝦	課	嘩	貨
326X	迦	過	霞	蚊	俄	峨	我	牙	画	臥	芽	蛾	賀	雅	餓	駕
327X	介	会	解	回	塊	壊	廻	快	怪	悔	恢	懐	戒	拐	改	
332X		魁	晦	械	海	灰	界	皆	絵	芥	蟹	開	階	貝	凱	劾
333X	外	咳	害	崖	慨	概	涯	碍	蓋	街	該	鎧	骸	浬	馨	蛙
334X	垣	柿	蠣	鈎	劃	嚇	各	廓	拡	攪	格	核	殻	獲	確	穫
335X	覚	角	赫	較	郭	閣	隔	革	学	岳	楽	額	顎	掛	笠	樫
336X	橿	梶	鰍	潟	割	喝	恰	括	活	渇	滑	葛	褐	轄	且	鰹
337X	叶	椛	樺	鞄	株	兜	竈	蒲	釜	鎌	噛	鴨	栢	茅	萱	
342X		粥	刈	苅	瓦	乾	侃	冠	寒	刊	勘	勧	巻	喚	堪	姦
343X	完	官	寛	干	幹	患	感	慣	憾	換	敢	柑	桓	棺	款	歓
344X	汗	漢	澗	灌	環	甘	監	看	竿	管	簡	緩	缶	翰	肝	艦
345X	莞	観	諫	貫	還	鑑	間	閑	関	陥	韓	館	舘	丸	含	岸
346X	巌	玩	癌	眼	岩	翫	贋	雁	頑	顔	願					

（例えば，可のコードは3244と読みます．実際の使用には"&H"をつけて「&H3244」とします．）

キ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
346X												企	伎	危	喜	器
347X	基	奇	嬉	寄	岐	希	幾	忌	揮	机	旗	既	期	棋	棄	
352X		機	帰	毅	気	汽	畿	祈	季	稀	紀	徽	規	記	貴	起
353X	軌	輝	飢	騎	鬼	亀	偽	儀	妓	宜	戯	技	擬	欺	犠	疑
354X	祇	義	蟻	誼	議	掬	菊	鞠	吉	吃	喫	桔	橘	詰	砧	杵
355X	黍	却	客	脚	虐	逆	丘	久	仇	休	及	吸	宮	弓	急	救
356X	朽	求	汲	泣	灸	球	究	窮	笈	級	糾	給	旧	牛	去	居
357X	巨	拒	拠	挙	渠	虚	許	距	鋸	漁	禦	魚	亨	享	京	
362X		供	侠	僑	兇	競	共	凶	協	匡	卿	叫	喬	境	峡	強
363X	彊	怯	恐	恭	挟	教	橋	況	狂	狭	矯	胸	脅	興	蕎	郷
364X	鏡	響	饗	驚	仰	凝	堯	暁	業	局	曲	極	玉	桐	粁	僅
365X	勤	均	巾	錦	斤	欣	欽	琴	禁	禽	筋	緊	芹	菌	衿	襟
366X	謹	近	金	吟	銀											

コードはすべて16進形式

ク

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
366X						九	倶	句	区	狗	玖	矩	苦	軀	駆	駈
367X	駒	具	愚	虞	喰	空	偶	寓	遇	隅	串	櫛	釧	屑	屈	
372X		掘	窟	沓	靴	轡	窪	熊	隈	粂	栗	繰	桑	鍬	勲	君
373X	薫	訓	群	軍	郡											

（例えば、君のコードは372Fと読みます．実際の使用は"&H"をつけて「&H327F」とします．）

ケ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
373X						卦	袈	祁	係	傾	刑	兄	啓	圭	珪	型
374X	契	形	径	恵	慶	慧	憩	掲	携	敬	景	桂	渓	畦	稽	系
375X	経	継	繋	罫	茎	荊	蛍	計	詣	警	軽	頚	鶏	芸	迎	鯨
376X	劇	戟	撃	激	隙	桁	傑	欠	決	潔	穴	結	血	訣	月	件
377X	倹	倦	健	兼	券	剣	喧	圏	堅	嫌	建	憲	懸	拳	捲	
382X		検	権	牽	犬	献	研	硯	絹	県	肩	見	謙	賢	軒	遣
383X	鍵	険	顕	験	鹸	元	原	厳	幻	弦	減	源	玄	現	絃	舷
384X	言	諺	限													

コ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
384X				乎	個	古	呼	固	姑	孤	己	庫	弧	戸	故	枯
385X	湖	狐	糊	袴	股	胡	菰	虎	誇	跨	鈷	雇	顧	鼓	五	互
386X	伍	午	呉	吾	娯	後	御	悟	梧	檎	瑚	碁	語	誤	護	醐
387X	乞	鯉	交	佼	侯	候	倖	光	公	功	効	勾	厚	口	向	
392X		后	喉	坑	垢	好	孔	孝	宏	工	巧	巷	幸	広	庚	康
393X	弘	恒	慌	抗	拘	控	攻	昂	晃	更	杭	校	梗	構	江	洪
394X	浩	港	溝	甲	皇	硬	稿	糠	紅	紘	絞	綱	耕	考	肯	肱
395X	腔	膏	航	荒	行	衡	講	貢	購	郊	酵	鉱	礦	鋼	閤	降
396X	項	香	高	鴻	剛	劫	号	合	壕	拷	濠	豪	轟	麹	克	刻
397X	告	国	穀	酷	鵠	黒	獄	漉	腰	甑	忽	惚	骨	狛	込	
3A2X		此	頃	今	困	坤	墾	婚	恨	懇	昏	昆	根	梱	混	痕
3A3X	紺	艮	魂													

コードはすべて16進形式

サ		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3A3X				些	佐	叉	唆	嵯	左	差	査	沙	瑳	砂	詐	鎖
	3A4X	裟	坐	座	挫	債	催	再	最	哉	塞	妻	宰	彩	才	採	栽
	3A5X	歳	済	災	采	犀	砕	砦	祭	斎	細	菜	裁	載	際	剤	在
	3A6X	材	罪	財	冴	坂	阪	堺	榊	肴	咲	崎	埼	碕	鷺	作	削
	3A7X	咋	搾	昨	朔	柵	窄	策	索	錯	桜	鮭	笹	匙	冊	刷	
	3B2X		察	拶	撮	擦	札	殺	薩	雑	皐	鯖	捌	錆	鮫	皿	晒
	3B3X	三	傘	参	山	惨	撒	散	桟	燦	珊	産	算	纂	蚕	讃	賛
	3B4X	酸	餐	斬	暫	残											

(例えば，山のコードは3 B33と読みます．実際の使用は "&H" をつけて「&H 3 B33」とします)

シ		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
	3B4X						仕	仔	伺	使	刺	司	史	嗣	四	士	始
	3B5X	姉	姿	子	屍	市	師	志	思	指	支	孜	斯	施	旨	枝	止
	3B6X	死	氏	獅	祉	私	糸	紙	紫	肢	脂	至	視	詞	詩	試	誌
	3B7X	諮	資	賜	雌	飼	歯	事	似	侍	児	字	寺	慈	持	時	
	3C2X		次	滋	治	爾	璽	痔	磁	示	而	耳	自	蒔	辞	汐	鹿
	3C3X	式	識	鴫	竺	軸	宍	雫	七	叱	執	失	嫉	室	悉	湿	漆
	3C4X	疾	質	実	蔀	篠	偲	柴	芝	屡	蕊	縞	舎	写	射	捨	赦
	3C5X	斜	煮	社	紗	者	謝	車	遮	蛇	邪	借	勺	尺	杓	灼	爵
	3C6X	酌	釈	錫	若	寂	弱	惹	主	取	守	手	朱	殊	狩	珠	種
	3C7X	腫	趣	酒	首	儒	受	呪	寿	授	樹	綬	需	囚	収	周	
	3D2X		宗	就	州	修	愁	拾	洲	秀	秋	終	繍	習	臭	舟	蒐
	3D3X	衆	襲	讐	蹴	輯	週	酋	酬	集	醜	什	住	充	十	従	戎
	3D4X	柔	汁	渋	獣	縦	重	銃	叔	夙	宿	淑	祝	縮	粛	塾	熟
	3D5X	出	術	述	俊	峻	春	瞬	竣	舜	駿	准	循	旬	楯	殉	淳
	3D6X	準	潤	盾	純	巡	遵	醇	順	処	初	所	暑	曙	渚	庶	緒
	3D7X	署	書	薯	藷	諸	助	叙	女	序	徐	恕	鋤	除	傷	償	
	3E2X		勝	匠	升	召	哨	商	唱	嘗	奨	妾	娼	宵	将	小	少
	3E3X	尚	庄	床	廠	彰	承	抄	招	掌	捷	昇	昌	昭	晶	松	梢
	3E4X	樟	樵	沼	消	渉	湘	焼	焦	照	症	省	硝	礁	祥	称	章
	3E5X	笑	粧	紹	肖	菖	蒋	蕉	衝	裳	訟	証	詔	詳	象	賞	醤

次頁につづく

コードはすべて16進形式

シ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3E6X	鉦	鍾	鐘	障	鞘	上	丈	丞	乗	冗	剰	城	場	壌	嬢	常
3E7X	情	擾	条	杖	浄	状	畳	穣	蒸	譲	醸	錠	嘱	埴	飾	
3F2X		拭	植	殖	燭	織	職	色	触	食	蝕	辱	尻	伸	信	侵
3F3X	唇	娠	寝	審	心	慎	振	新	晋	森	榛	浸	深	申	疹	真
3F4X	神	秦	紳	臣	芯	薪	親	診	身	辛	進	針	震	人	仁	刃
3F5X	塵	壬	尋	甚	尽	腎	訊	迅	陣	靱						

(例えば、上のコードは3E65と読みます．実際の使用は"&H"をつけて「&H3E65」とします)

ス

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
3F5X											笥	諏	須	酢	図	厨
3F6X	逗	吹	垂	帥	推	水	炊	睡	粋	翠	衰	遂	酔	錐	錘	随
3F7X	瑞	髄	崇	嵩	数	枢	趨	雛	据	杉	椙	菅	頗	雀	裾	
402X		澄	摺	寸												

セ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
402X					世	瀬	畝	是	凄	制	勢	姓	征	性	成	政
403X	整	星	晴	棲	栖	正	清	牲	生	盛	精	聖	声	製	西	誠
404X	誓	請	逝	醒	青	静	斉	税	脆	隻	席	惜	戚	斥	昔	析
405X	石	積	籍	績	脊	責	赤	跡	蹟	碩	切	拙	接	摂	折	設
406X	窃	節	説	雪	絶	舌	蝉	仙	先	千	占	宣	専	尖	川	戦
407X	扇	撰	栓	栴	泉	浅	洗	染	潜	煎	煽	旋	穿	箭	線	
412X		繊	羨	腺	舛	船	薦	詮	賎	践	選	遷	銭	銑	閃	鮮
413X	前	善	漸	然	全	禅	繕	膳	糎							

ソ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
413X										噌	塑	岨	措	曾	曽	楚
414X	狙	疏	疎	礎	祖	租	粗	素	組	蘇	訴	阻	遡	鼠	僧	創
415X	双	叢	倉	喪	壮	奏	爽	宋	層	匝	惣	想	捜	掃	挿	掻
416X	操	早	曹	巣	槍	槽	漕	燥	争	痩	相	窓	糟	総	綜	聡
417X	草	荘	葬	蒼	藻	装	走	送	遭	鎗	霜	騒	像	増	憎	
422X		臓	蔵	贈	造	促	側	則	即	息	捉	束	測	足	速	俗

次頁につづく

コードはすべて16進形式

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ソ	423X	属	賊	族	続	卒	袖	其	揃	存	孫	尊	損	村	遜		

（例えば，存のコードは4238と読みます．実際の使用は "&H" をつけて「&H4238」とします）

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
タ	423X															他	多
	424X	太	汰	詑	唾	堕	妥	惰	打	柁	舵	楕	陀	駄	騨	体	堆
	425X	対	耐	岱	帯	待	怠	態	戴	替	泰	滞	胎	腿	苔	袋	貸
	426X	退	逮	隊	黛	鯛	代	台	大	第	醍	題	鷹	滝	瀧	卓	啄
	427X	宅	托	択	拓	沢	濯	琢	託	鐸	濁	諾	茸	凧	蛸	只	
	432X		叩	但	達	辰	奪	脱	巽	竪	辿	棚	谷	狸	鱈	樽	誰
	433X	丹	単	嘆	坦	担	探	旦	歎	淡	湛	炭	短	端	箪	綻	耽
	434X	胆	蛋	誕	鍛	団	壇	弾	断	暖	檀	段	男	談			

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
チ	434X														値	知	地
	435X	弛	恥	智	池	痴	稚	置	致	蜘	遅	馳	築	畜	竹	筑	蓄
	436X	逐	秩	窒	茶	嫡	着	中	仲	宙	忠	抽	昼	柱	注	虫	衷
	437X	註	酎	鋳	駐	樗	瀦	猪	苧	著	貯	丁	兆	凋	喋	寵	
	442X		帖	帳	庁	弔	張	彫	徴	懲	挑	暢	朝	潮	牒	町	眺
	443X	聴	脹	腸	蝶	調	諜	超	跳	銚	長	頂	鳥	勅	捗	直	朕
	444X	沈	珍	賃	鎮	陳											

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ツ	444X						津	墜	椎	槌	追	鎚	痛	通	塚	栂	掴
	445X	槻	佃	漬	柘	辻	蔦	綴	鍔	椿	潰	坪	壷	嬬	紬	爪	吊
	446X	釣	鶴														

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
テ	446X			亭	低	停	偵	剃	貞	呈	堤	定	帝	底	庭	廷	弟
	447X	悌	抵	挺	提	梯	汀	碇	禎	程	締	艇	訂	諦	蹄	逓	
	452X		邸	鄭	釘	鼎	泥	摘	擢	敵	滴	的	笛	適	鏑	溺	哲

次頁につづく

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
テ	453X	徹	撤	轍	迭	鉄	典	填	天	展	店	添	纏	甜	貼	転	顛
	454X	点	伝	殿	澱	田	電										

（例えば，天のコードは 4537 と読みます．実際の使用は "&H" をつけて「&H4537」とします．）

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ト	454X							兎	吐	堵	塗	妬	屠	徒	斗	杜	渡
	455X	登	菟	賭	途	都	鍍	砥	礪	努	度	土	奴	怒	倒	党	冬
	456X	凍	刀	唐	塔	塘	套	宕	島	嶋	悼	投	搭	東	桃	檮	棟
	457X	盗	淘	湯	濤	灯	燈	当	痘	禱	等	答	筒	糖	統	到	
	462X		董	蕩	藤	討	謄	豆	踏	逃	透	鐙	陶	頭	騰	闘	働
	463X	動	同	堂	導	憧	撞	洞	瞳	童	胴	萄	道	銅	峠	鴇	匿
	464X	得	徳	瀆	特	督	禿	篤	毒	独	読	栃	橡	凸	突	椴	届
	465X	鳶	苫	寅	酉	瀞	噸	屯	惇	敦	沌	豚	遁	頓	呑	曇	鈍

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ナ	466X	奈	那	内	乍	凪	薙	謎	灘	捺	鍋	楢	馴	縄	畷	南	楠
	467X	軟	難	汝													

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ニ	467X				二	尼	弐	迩	匂	賑	肉	虹	廿	日	乳	入	
	472X		如	尿	韮	任	妊	忍	認								

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ヌ	472X									濡							

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ネ	472X										禰	祢	寧	葱	猫	熱	年
	473X	念	捻	撚	燃	粘											

ノ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
473X						乃	廼	之	埜	嚢	悩	濃	納	能	脳	膿
474X	農	覗	蚤													

（例えば，能のコードは 473D と読みます．実際の使用は "&H" をつけて「&H473D」とします）

ハ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
474X				巴	把	播	覇	杷	波	派	琶	破	婆	罵	芭	馬
475X	俳	廃	拝	排	敗	杯	盃	牌	背	肺	輩	配	倍	培	媒	梅
476X	楳	煤	狽	買	売	賠	陪	這	蝿	秤	矧	萩	伯	剥	博	拍
477X	柏	泊	白	箔	粕	舶	薄	迫	曝	漠	爆	縛	莫	駁	麦	
482X		函	箱	硲	箸	肇	筈	櫨	幡	肌	畑	畠	八	鉢	溌	発
483X	醗	髪	伐	罰	抜	筏	閥	鳩	噺	塙	蛤	隼	伴	判	半	反
484X	叛	帆	搬	斑	板	氾	汎	版	犯	班	畔	繁	般	藩	販	範
485X	釆	煩	頒	飯	挽	晩	番	盤	磐	蕃	蛮					

ヒ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
485X												匪	卑	否	妃	庇
486X	彼	悲	扉	批	披	斐	比	泌	疲	皮	碑	秘	緋	罷	肥	被
487X	誹	費	避	非	飛	樋	簸	備	尾	微	枇	毘	琵	眉	美	
492X		鼻	柊	稗	匹	疋	髭	彦	膝	菱	肘	弼	必	畢	筆	逼
493X	桧	姫	媛	紐	百	謬	俵	彪	標	氷	漂	瓢	票	表	評	豹
494X	廟	描	病	秒	苗	錨	鋲	蒜	蛭	鰭	品	彬	斌	浜	瀕	貧
495X	賓	頻	敏	瓶												

フ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
495X					不	付	埠	夫	婦	富	冨	布	府	怖	扶	敷
496X	斧	普	浮	父	符	腐	膚	芙	譜	負	賦	赴	阜	附	侮	撫
497X	武	舞	葡	蕪	部	封	楓	風	葺	蕗	伏	副	復	幅	服	
4A2X		福	腹	複	覆	淵	弗	払	沸	仏	物	鮒	分	吻	噴	墳
4A3X	憤	扮	焚	奮	粉	糞	紛	雰	文	聞						

コードはすべて 16 進形式

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
へ	4A3X											丙	併	兵	塀	幣	平
	4A4X	弊	柄	並	蔽	閉	陛	米	頁	僻	壁	癖	碧	別	瞥	蔑	箆
	4A5X	偏	変	片	篇	編	辺	返	遍	便	勉	娩	弁	鞭			

（例えば，平のコードは 4A3F と読みます．実際の使用は "&H" をつけて「&H4A3F」とします）

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ホ	4A5X														保	舗	鋪
	4A6X	圃	捕	歩	甫	補	輔	穂	募	墓	慕	戊	暮	母	簿	菩	倣
	4A7X	俸	包	呆	報	奉	宝	峰	峯	崩	庖	抱	捧	放	方	朋	
	4B2X		法	泡	烹	砲	縫	胞	芳	萌	蓬	蜂	褒	訪	豊	邦	鋒
	4B3X	飽	鳳	鵬	乏	亡	傍	剖	坊	妨	帽	忘	忙	房	暴	望	某
	4B4X	棒	冒	紡	肪	膨	謀	貌	貿	鉾	防	吠	頬	北	僕	卜	墨
	4B5X	撲	朴	牧	睦	穆	釦	勃	没	殆	堀	幌	奔	本	翻	凡	盆

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
マ	4B6X	摩	磨	魔	麻	埋	妹	昧	枚	毎	哩	槙	幕	膜	枕	鮪	柾
	4B7X	鱒	桝	亦	俣	又	抹	末	沫	迄	侭	繭	麿	万	慢	満	
	4C2X		漫	蔓													

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ミ	4C2X				味	未	魅	巳	箕	岬	密	蜜	湊	蓑	稔	脈	妙
	4C3X	粍	民	眠													

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
ム	4C3X				務	夢	無	牟	矛	霧	鵡	椋	婿	娘			

		0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
メ	4C3X														冥	名	命
	4C4X	明	盟	迷	銘	鳴	姪	牝	滅	免	棉	綿	緬	面	麺		

モ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C4X															摸	模
4C5X	茂	妄	孟	毛	猛	盲	網	耗	蒙	儲	木	黙	目	杢	勿	餅
4C6X	尤	戻	籾	貰	問	悶	紋	門	匁							

（例えば、門のコードは4C67と読みます。実際の使用は"&H"をつけて「&H4C67」とします）

ヤ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C6X										也	冶	夜	爺	耶	野	弥
4C7X	矢	厄	役	約	薬	訳	躍	靖	柳	薮	鑓					

ユ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4C7X												愉	愈	油	癒	
4D2X		諭	輸	唯	佑	優	勇	友	宥	幽	悠	憂	揖	有	柚	湧
4D3X	涌	猶	猷	由	祐	裕	誘	遊	邑	郵	雄	融	夕			

ヨ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4D3X														予	余	与
4D4X	誉	輿	預	傭	幼	妖	容	庸	揚	揺	擁	曜	楊	様	洋	溶
4D5X	熔	用	窯	羊	耀	葉	蓉	要	謡	踊	遥	陽	養	慾	抑	欲
4D6X	沃	浴	翌	翼	淀											

ラ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4D6X						羅	螺	裸	来	莱	頼	雷	洛	絡	落	酪
4D7X	乱	卵	嵐	欄	濫	藍	蘭	覧								

リ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4D7X									利	吏	履	李	梨	理	璃	
4E2X		痢	裏	裡	里	離	陸	律	率	立	葎	掠	略	劉	流	溜
4E3X	琉	留	硫	粒	隆	竜	龍	侶	慮	旅	虜	了	亮	僚	両	凌
4E4X	寮	料	梁	涼	猟	療	瞭	稜	糧	良	諒	遼	量	陵	領	力
4E5X	緑	倫	厘	林	淋	燐	琳	臨	輪	隣	鱗	麟				

コードはすべて16進形式

ル

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4E5X													瑠	塁	涙	累
4E6X	類															

（例えば，類のコードは4E60と読みます．実際の使用は"&H"をつけて「&H4E60」とします）

レ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4E6X		令	伶	例	冷	励	嶺	怜	玲	礼	苓	鈴	隷	零	霊	麗
4E7X	齢	暦	歴	列	劣	烈	裂	廉	恋	憐	漣	煉	簾	練	聯	
4F2X		蓮	連	錬												

ロ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4F2X					呂	魯	櫓	炉	賂	路	露	労	婁	廊	弄	朗
4F3X	楼	榔	浪	漏	牢	狼	籠	老	聾	蝋	郎	六	麓	禄	肋	録
4F4X	論															

ワ

	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
4F4X		倭	和	話	歪	賄	脇	惑	枠	鷲	亙	亘	鰐	詫	藁	蕨
4F5X	椀	湾	碗	腕												

付録5 F-BASICのエラーメッセージ

エラーコード	エラーメッセージ	内容
01	Next Without For	NEXTに対応するFOR文がない.
02	Syntax Error	コマンドまたは, 文の書き方に誤りがある.
03	Return Without Gosub	GOSUB文によって呼出されていないのに, RETURN文に出会った.
04	Out Of Data	READ文によって読込むべきデータがない.
05	Illegal Function Call	関数やステートメントの呼び方に誤りがある.
06	Overflow	整数値または実数値が, 許される範囲をこえている. または代入される数値が大きすぎる. 整数値のとき −32768~32767 の範囲にない. 実数値のとき −1.70141E+38~1.70141E+38 の範囲にない.
07	Out Of Memory	メモリが足りなくなった.
08	Undefined Line Number	指定された行番号が定義されていない.
09	Subscript Out Of Range	配列の添字が0から上限の範囲にない.
10	Duplicate Definition	同じ名前の配列または, ユーザ関数を2度宣言している.
11	Division By Zero	除算の分母が0である.
12	Illegal Direct	直接モードで使えないステートメントを用いた.
13	Type Mismatch	変数または定数の型が合わない. 文字と数値を演算しようとしている. 代入の左辺と右辺, 関数の引数の型, ……
14	Out Of String Space	文字領域が足りなくなった.
15	String Too Long	文字定数が256文字をこえている. または文字式の結果が256文字以上になった.
16	String Formula Too Complex	文字式が複雑すぎる. 文字式のネストの深さは10まで.
17	Can't Continue	CONTコマンドによるプログラムの続行ができない.
18	Undefined User Function	定義されていない関数を参照している.
19	NO Resume	エラー処理ルーチンにRESUMEがない.
20	Resume Without Error	エラーが起きていないのに, RESUME文を実行しようとした.
21	Unprintable Error	エラーメッセージの定義されていないエラーを出そうとした.
22	Missing Operand	必要なオペランドが抜けている.
23	For Without Next	FOR~NEXTの対応が正しくない.
24	While without Wend	WHILE文に対応するWEND文がない.
25	Wend without While	WEND文に対応するWHILE文がない.
26	Bubble Full	バブルカセットがいっぱいであり, データの登録ができない.
50	Bad File Number	オープンされていないファイル番号を使用した.
51	Bad File Mode	入力モードでオープンしたファイル番号に対して出力しようとした. または, 出力モードでオープンしたファイル番号から入力しようとした.
52	File Already Open	ファイルを二重にオープンしようとした.

エラーコード	エラーメッセージ	内　　　容
53	Device I/O Error	使用したデバイスに入出力エラーが発生した.
54	Input Past End	ファイルの全てのデータを読んだ後に, INPUT文を実行した.
55	Bad File Descriptor	ファイルディスクリプタの記述に誤りがある.
56	Direct Statement In File	アスキー形式のプログラムファイル中に, 直接ステートメントがあった.
57	File Not Open	ファイルがオープンされていない.
58	Bad Data In File	ファイル上のデータの形式が間違っている.
59	Device In Use	使用中のデバイスに対して, 再度オープンしようとした.
60	Device Unavailable	I/O デバイスが入出力可能な状態にない.
61	Buffer Overflow	入出力バッファがオーバーフローした.
62	Protected Program	保護されているプログラムに, 書込み修正を行おうとした.
63	File Not Found	指定されたファイル名が見つからない.
64	File Already Exists	指定されたファイル名はすでに存在している.
65	Directory Full	ディレクトリ領域がいっぱいであり, 新たなファイルの登録ができない.
66	Too Many Open Disk Files	確保されているファイルの個数をこえてオープンしようとした.
67	Disk Full	ディスクがいっぱいであり, データの登録ができない.
68	Field Overflow	フィールドの長さが256バイトをこえている.
69	String Not Fielded	Field文で宣言された文字変数以外の変数にLSET, RSETを用いて代入しようとしている.
70	Bad Record Number	指定されたレコード番号は存在しない.
71	Bad File Structure	ファイルの構成に誤りがある.
72	Drive Not Ready	指定されたドライブ番号はReady状態にはない.
73	Disk Write Protected	Diskが書込み保護されている.

備考：エラーコード65〜73は, Diskモードに関するエラーメッセージです.

付録6　F-BASIC 命令一覧表

○印…使用可，X印…使用不可

種別	命令	内容	バージョン		
			V1.0	V2.0	V3.0
コマンド	AUTO	行の先頭に行番号を自動的に発生する．	○	○	○
	DELETE	プログラムの行を削除する．	○	○	○
	LIST	メモリ内にあるプログラムの全部または，一部をCRTなどに出力する．	○	○	○
	LLIST	メモリ内にあるプログラムの全部または，一部をプリンタへ出力する．	×	○	○
	UNLIST	非表示行番号を指定します．	○	○	○
	RENUM	プログラムの各行の行番号を付け直す．	○	○	○
	NEW	メモリにあるプログラムを消去し，すべての変数を初期化する	○	○	○
	CLEAR	全ての数値変数を0に，全ての文字変数を空文字列に初期設定し，オペランドの指定によりBASICの使用する上限を設定する．	○	○	○
	CONT	BREAKキー（V1.0 V2.0ではSTOPキー）入力後またはSTOP，END文実行後，停止しているプログラムの実行を再開する．	○	○	○
	RUN	メモリまたは，ファイルにあるプログラムを実行する．	○	○	○
	LOAD	プログラムファイルをメモリにロードする．	○	○	○
	LOAD?	カセットテープのファイルの内容とチェックサムの照合を行う．	○	○	○
	SAVE	メモリ内にあるプログラムをファイル格納する．	○	○	○
	FILES	指定されたデバイスのディレクトリ・リストを出力する．	○	○	○
	NAME	フロッピィディスク上のファイル名を変更する．	○	○	○
	KILL	フロッピィディスクまたはバブルカセット上のファイルを削除する．	○	○	○
	MERGE	メモリにあるプログラムと，指定されたファイルのプログラムを混ぜ合わせる．	○	○	○
	SKIPF	指定したファイルの次にカセットテープを進める．	○	○	○
	DSKINI	フロッピィディスクのディレクトリの初期化を行う．	○	○	○
	BUBINI	バブルカセットのディレクトリの初期化を行う．	○	○	×
	EXEC	機械語プログラムを実行する．	○	○	○
	LOADM	機械語プログラム・ファイルをメモリにロードし，実行する．	○	○	○
	SAVEM	メモリ内の機械語プログラムをファイルにセーブする．	○	○	○

○印…使用可，×印…使用不可

種別	命令	内容	バージョン		
			V1.0	V2.0	V3.0
コマンド	HARDC	画面データをプリンタに出力する．	○	○	○
	MON	BASICコマンド・モードからモニタに移る． M：メモリの内容を変更する． G：指定アドレスに分岐する． R：レジスタの内容を表示し，変更を可能にする． D：指定アドレスから64バイトの内容を表示する．	○	○	○
	TERM	動作モードをターミナルモードにする．	○	○	○
	EDIT	指定された行を画面に表示する．	○	○	○
一般ステートメント	DEF FN	ユーザの関数を定義し，それに名前を付ける．	○	○	○
	DEF USR	機械語プログラムの開始番号を指定する．	○	○	○
	DEF INT	変数の型を整数に宣言する．	○	○	○
	DEF SNG	変数の型を単精度に宣言する．	○	○	○
	DEF DBL	変数の型を倍精度に宣言する．	○	○	○
	DEF STR	変数の型を文字に宣言する．	○	○	○
	REM	プログラムの中の注釈である（▼で代用可能）．	○	○	○
	END	プログラムの実行を終了し，全てのオープンされているファイルをクローズした後，コマンドレベル待ちになる．	○	○	○
	FOR～NEXT	一連の命令を繰返し実行する．	○	○	○
	GOSUB	サブルーチンを呼出し，サブルーチン終了後はGOSUB文の直後の文に戻る．	○	○	○
	GOTO	無条件に指定された行番号に分岐する．	○	○	○
	ON～GOTO	式の値により，指定された行番号の一つに分岐する．	○	○	○
	RETURN	サブルーチンを終了し，呼出したプログラムへ復帰する．	○	○	○
	STOP	プログラムの実行を停止してコマンド待ちになる．	○	○	○
	ON～GOSUB	式の値により，指定された行番号を持つサブルーチンを呼出す	○	○	○
	IF～THEN～ELSE	式の値により，実行すべき文を選択する．	○	○	○
	WHILE～WEND	一連の命令を条件付きで繰返し実行する．	○	○	○
	LET	右辺の式の結果を左辺の変数に代入する．	○	○	○
	SWAP	二つの変数の値を交換する．	○	○	○
	DIM	配列変数の次元数と添字の最大値を指定し，その変数にメモリ領域を割当てる．	○	○	○

○印…使用可，×印…使用不可

種別	命令	内容	バージョン		
			V1.0	V2.0	V3.0
一般ステートメント	POKE	メモリの指定番地にデータを書込む．	○	○	○
	DATA	READ文によって読込まれる数値および，文字定数を格納する．	○	○	○
	READ	DATA文で定義した定数を変数に読込む．	○	○	○
	RESTORE	DATA文を最初から読むように指示する．	○	○	○
	LSET，RSET	文字データをランダムバッファに移す．	○	○	○
	RANDOMIZE	乱数の系列を変更する．	○	○	○
	ERROR	BASICのエラー発生をシミュレートしたり，ユーザのエラー番号の定義を可能にする．	○	○	○
	ON ERROR GOTO	エラートラップ機能を可能にする．	○	○	○
	RESUME	エラー処理終了後，プログラムの実行を再開する．	○	○	○
	BEEP	内蔵スピーカによりブザーを鳴らす．	○	○	○
	MOTOR	カセットテープレコーダのモータを制御する．	○	○	○
	TRON	プログラム実行状態を追跡する．	○	○	○
	TROFF	プログラム実行状態の追跡を止める．	○	○	○
	CHAIN	メモリ上のプログラムから指定したプログラムへ変数を引き渡し，実行する．	×	○	○
	COMMON	CHAIN文により，連結されるプログラムへ引き渡す変数を指定する．	×	○	○
	ERASE	変数および配列変数をプログラムから消去する．	×	○	○
入出力ステートメント	INPUT	キーボートから入力するデータを読取る．	○	○	○
	LINE INPUT	1桁全体の文字列（255文字以内）を区切ることなく文字変数に読込む．	○	○	○
	PRINT	画面に式の評価結果を出力する（？で代用可能）．	○	○	○
	LPRINT	データをプリンタへ出力する．	×	○	○
	PRINT@	漢字を画面に表示する．	○	○	○
	PRINT USING	文字または数値を指定した書式で画面に出力する．	○	○	○
	LPRINT USING	文字または数値を指定した書式でプリンタへ出力する．	×	○	○
	OPEN	ファイルのオープン処理を行う．	○	○	○
	CLOSE	ファイルのクローズ処理を行う．	○	○	○
	INPUT#	ファイルからデータを読込み，変数に代入する．	○	○	○

○印…使用可，×印…使用不可

種別	命　　令	内　　　　容	バージョン		
			V1.0	V2.0	V3.0
入出力ステートメント	PRINT#	式の評価結果を指定されたファイルに出力する．	○	○	○
	PRINT# USING	式の評価結果を，指定した書式でファイルに出力する．	○	○	○
	LINE INPUT#	指定されたファイルから1行を区切ることなく読込み，変数に代入する．	○	○	○
	FIELD	ランダムファイルのバッファに変数の領域を割当てる．	○	○	○
	GET	ランダムファイルから指定されたレコードを，バッファに読込む．	○	○	○
	PUT	バッファの内容を指定されたランダムファイルに書込む．	○	○	○
	DSKO$	システムランダムバッファの内容を，指定されたセクターに書込む．	○	○	○
	BUBW	変数または配列の内容を，バブルカセットの指定されたページに書込む．	○	○	×
	BUBR	バブルカセットの指定されたページの内容を，変数または配列に読込む．	○	○	×
画面制御・グラフィック機能	WIDTH	画面に表示する文字の行数と桁数を指定する．	○	○	○
	CONSOLE	スクロール　ウインドの大きさを指定する．	○	○	○
	COLOR	画面に表示する文字，グラフィックの色および，背景色を指定する．	○	○	○
	COLOR	パレットコードを指定する．	×	×	○
	SCREEN	アクティブVRAMコードとディスプレイVRAMコードを指定する．	×	×	○
	CLS	指定画面をクリアし，カーソルをその画面のホーム・ポジションに移す．	○	○	○
	LOCATE	CRT画面上の任意の位置にカーソルを移動する．	○	○	○
	PSET	画面上の任置にドットを設定する．	○	○	○
	PRESET	画面上の任意の位置のドットを背景色にする．	○	○	○
	LINE	画面上にキャラクタ，ドットを使って線および箱を表示する．	○	○	○
	CONNECT	指定座標間を直線で結ぶ．	○	○	○
	SYMBOL	画面上の任意の位置に文字列を指定角度，指定サイズで表示する．	○	○	○
	GET@	画面上の任意の領域のキャラクタ，ドットパターンを配列に読込む．	○	○	○

○印…使用可，×印…使用不可

種別	命　　令	内　　容	バージョン		
			V1.0	V2.0	V3.0
	PUT@	GET@で読込まれたキャラクタ，ドットパターンを任意の位置に表示する．	○	○	○
	CIRCLE	画面上の任意の座標を中心点として，円または円弧を描く．	○	○	○
	GCURSOR	画面上のグラフィック・カーソルのドット座標を読取る．	○	○	○
	PAINT	指定された境界色で囲まれた範囲を塗りつぶす．	○	○	○
音楽演奏機能	PLAY	音楽の演奏を行う	×	×	○
	SOUND	PSG（Programmable Sound Generater）を直接，コントロールする．	×	×	○
プログラマブル・ファンクションキー機能	KEY	プログラマブル・ファンクションキーに文字列を定義する．	○	○	○
	KEY LIST	プログラマブル・ファンクションキーに定義されている文字列を，画面に表示する．	○	○	○
	KEY(n) ON／OFF／STOP	プログラマブル・ファンクションキーからの割込みの許可，禁止，停止をする．	○	○	○
	ON KEY(n) GOSUB	プログラマブル・ファンクションキーからの割込みルーチンの定義をする．	○	○	○
タイマ割込み機能	ON TIME GOSUB	タイマ割込みルーチンの定義をする．	○	○	○
	TIME	タイマ割込みの時刻を設定する．	○	○	○
	TIME ON／OFF／STOP	タイマ割込みの許可，禁止，停止をする．	○	○	○
	ON INTERVAL GOSUB	インターバルタイマ割込みルーチンの定義をする．	○	○	○
	INTERVAL	インターバルタイマ割込みの間隔を指定する．	○	○	○
	INTERVAL ON／OFF／STOP	インターバルタイマ割込みの許可，禁止，停止をする．	○	○	○
回線制御機能	ON COM(n) GOSUB	通信回線からの入力割込み時の処理ルーチンの定義をする．	○	○	○
	COM(n) ON／OFF／STOP	通信回線からの入力割込みの許可，禁止，停止をする．	○	○	○
	OPEN	通信回線の入出力を可能にするため，ファイルをオープンスる．	○	○	○
	CLOSE	通信回線に割当てられたファイルをクローズする．	○	○	○
	INPUT#	通信回線からデータを入力する．	○	○	○
	LINE INPUT#	通信回線から1行のデータを入力する．	○	○	○
	PRINT#	通信回線にデータを出力する．	○	○	○
	LIST	通信回線にプログラムリストを出力する．	○	○	○

○印…使用可，×印…使用不可

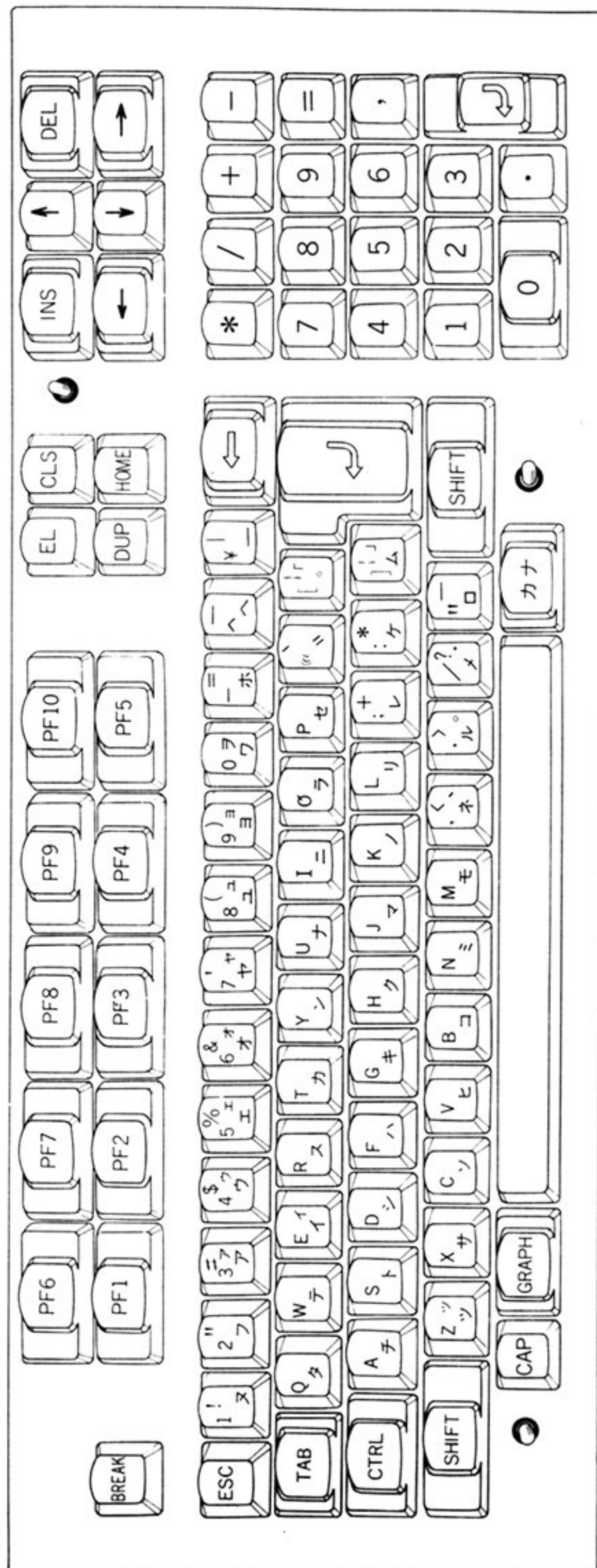
種別	命　令	内　容	バージョン		
			V1.0	V2.0	V3.0
数値関数	ABS	絶対値を与える．	○	○	○
	ATN	三角関数アークタンジェントの値を与える．	○	○	○
	COS	三角関数コサインの値を与える．	○	○	○
	EXP	eを底として指数関数を与える．	○	○	○
	FIX	引数の値の整数部分を与える．	○	○	○
	INT	引数の値を超えない最大の整数を与える．	○	○	○
	LOG	自然対数の値を与える．	○	○	○
	RND	0と1の間の乱数を与える．	○	○	○
	SGN	引数が正の場合は1，0の場合は0，負の場合は－1を与える．	○	○	○
	SIN	三角関数サインの値を与える．	○	○	○
	SQR	平方根を与える．	○	○	○
	TAN	三角関数タンジェントの値を与える．	○	○	○
	CSNG	式の値を単精度形式の数値に変換する．	○	○	○
	CDBL	式の値を倍精度形式の数値に変換する．	○	○	○
	CINT	小数部分を四捨五入して整数に変換する．	○	○	○
ストリング関数	CHR$	引数の値に対応する文字を与える．	○	○	○
	HEX$	整数を16進数の文字列に変換する．	○	○	○
	LEFT$	文字列の左から指定された数の文字列を取出す．	○	○	○
	MID$	指定された文字位置から任意の文字数を取出す．	○	○	○
	OCT$	整数を8進数の文字列に変換する．	○	○	○
	RIGHT$	文字列の右から指定された数の文字列を取出す．	○	○	○
	SPACE$	指定された数の空白よりなる文字列を与える．	○	○	○
	STR$	数値を表わす文字列を与える．	○	○	○
	STRING$	指定された文字だけで作られた文字列を与える．	○	○	○
	ASC	指定した文字列の最初の文字のキャラクタコードを与える．	○	○	○
	INSTR	指定された文字列を探索し，その位置を与える．	○	○	○
	LEN	文字列の長さを与える．	○	○	○
	VAL	文字列の表わす数値を与える．	○	○	○

○印…使用可，×印…使用不可

種別	命令	内容	バージョン		
			V1.0	V2.0	V3.0
一般関数	CSRLIN	画面上のカーソルの垂直位置を与える．	○	○	○
	POS	カーソル，プリンタヘッドの水平位置を与える．	○	○	○
	LPOS	プリンタヘッドの水平位置を与える．	×	○	○
	POINT	指定した位置にドットのセット状態を与える．	○	○	○
	ERR	エラー発生時のエラー番号を与える．	○	○	○
	ERL	エラー発生時の行番号を与える．	○	○	○
	VARPTR	変数名で指定されるデータの格納されている先頭番地を与える．	○	○	○
	USR	引数よりユーザのアセンブリ言語ルーチンを呼出す．	○	○	○
	PEEK	メモリの内容を読出す．	○	○	○
	FRE	メモリの未使用領域を与える．	○	○	○
	TIME$	内蔵タイマの示す時刻を与える．	○	○	○
	TIME	00：00：00を基準とする秒単位の時刻を示す．	○	○	○
	DATE$	内蔵タイマの示す日付けを与える．	○	○	○
	DATE	1月1日を基準とするトータル日数を示す．	○	○	○
	SPC	指定された数の空白をプリントする．	○	○	○
	TAB	指定された桁位置まで空白をプリントする．	○	○	○
	SCREEN	指定座標のキャラクタコードおよび属性を与える．	○	○	○
入出力関数	CVI	2バイトの文字列を数値データに変換する．	○	○	○
	CVS	4バイトの文字列を数値データに変換する．	○	○	○
	CVD	8バイトの文字列を数値データに変換する．	○	○	○
	MKI$	整数表記の数値を文字に変換する．	○	○	○
	MKS$	単精度表記の数値を文字に変換する．	○	○	○
	MKD$	倍精度表記の数値を文字に変換する．	○	○	○
	EOF	ファイルの終りを検出する．	○	○	○
	LOF	入力バッファ中の文字数を与える．	○	○	○
	LOC	ランダムファイルの次にGETまたは，PUTされるレコード番号を与える．	○	○	○
	DSKI$	指定されたセクタの内容をシステムランダムバッファに読込む	○	○	○

○印…使用可，×印…使用不可

種別	命　　令	内　　　容	バージョン		
			V1.0	V2.0	V3.0
	DSKF	ディスクの未使用領域のクラスタ数を示す．	○	○	○
	INPUT$	キーボートおよびファイルから指定された数の文字列を入力する．	○	○	○
	INKEY$	キーボードが押されていれば，その文字を与える．押されていなければ空文字を与える．	○	○	○
	ANPORT	アナログポートからA／D変換されたデータを読取る．	○	○	×

付録7　キーボード図

キー配列（通常モード）

キー配列（グラフィックモード）

索 引

T

U

V

W